Correction algorithmes

Pack et mise en page par Laurae, rédigé par critor sur le forum, administrateurs de TI-Planet.org

Introduction : C'est du gâteau !

Vous trouverez dans ce PDF tous les corrigés des exercices sur les algorithmes du BAC, mais uniquement ceux qui sont corrigés sur le forum par critor.

Documents: Annales des sujets inédits des nouveaux BAC et DNB (Brevet) 2013 corrigés:	**Programmes TI-Nspire**	**Programmes TI-76/82/83/84**
Terminale S Terminale ES Terminale L	Annales BAC S 2013 *(mViewer)*	
Terminale STG Terminale ST2S Terminale STI2D Terminale STD2A	SD2 *(dérivées pas à pas)* SINT *(intégration pas à pas)* SIPP *(intégration par parties pas à pas)* mCAS *(calcul exact)* Trigor *(cercle trigo)*	AutoCalc *(calcul exact)* Dynatrig *(cercle trigo)* Binomall *(loi binomiale)*
Première S Première ES Première L Première Technologique	mViewer *(lecteur images)* Nover *(overclocking)*	
Troisième Générale/Collège		

Pack et mise en page par Laurae, rédigé par critor sur le forum, administrateurs de TI-Planet.org

Sommaire

(cliquez sur le numéro de page pour accéder au corrigé voulu, mais aussi sur les titres en dehors du sommaire pour être redirigé sur TI-Planet)

Table des matières

Pack et mise en page par Laurae, rédigé par critor sur le forum, administrateurs de TI-Planet.org

Correction 1 : Eléments de correction exo d'algo BAC S maths Inde 2012

de **critor** » 19 Avr 2012 12:46

Dans une news précédente, nous te révélions suite au 1er sujet de maths du BAC S 2012 tombé hier en Inde que l'épreuve de cette année pouvait bien porter sur les ajouts du nouveau programme, et notamment l'algorithmique et la logique.

EXERCICE 1 (6 points)

Commun à tous les candidats

Les deux parties sont indépendantes.

Partie A

Un groupe de 50 coureurs, portant des dossards numérotés de 1 à 50, participe à une course cycliste qui comprend 10 étapes, et au cours de laquelle aucun abandon n'est constaté.
À la fin de chaque étape, un groupe de 5 coureurs est choisi au hasard pour subir un contrôle anti-dopage. Ces désignations de 5 coureurs à l'issue de chacune des étapes sont indépendantes. Un même coureur peut donc être contrôlé à l'issue de plusieurs étapes.

1. À l'issue de chaque étape, combien peut-on former de groupes différents de 5 coureurs ?
2. On considère l'algorithme ci-dessous dans lequel :
   - « rand(1, 50) » permet d'obtenir un nombre entier aléatoire appartenant à l'intervalle [1 ; 50]
   - l'écriture « $x := y$ » désigne l'affectation d'une valeur y à une variable x.

Variables	a, b, c, d, e sont des variables du type entier
Initialisation	$a := 0$; $b := 0$; $c := 0$; $d := 0$; $e := 0$
Traitement	Tant que $(a = b)$ ou $(a = c)$ ou $(a = d)$ ou $(a = e)$ ou $(b = c)$ ou $(b = d)$ $(b = e)$ ou $(c = d)$ ou $(c = e)$ ou $(d = e)$ Début du tant que $a :=$rand(1, 50) ; $b :=$ rand(1, 50) ; $c :=$and(1, 50) ; $d :=$ rand(1, 50) ; $e :=$ rand(1, 50) Fin du tant que
Sortie	Afficher a, b, c, d, e

   (a) Parmi les ensembles de nombres suivants, lesquels ont pu être obtenus avec cet algorithme :
   $L_1 = \{2, 11, 44, 2, 15\}$; $L_2 = \{8, 17, 41, 34, 6\}$;
   $L_3 = \{12, 17, 23, 17, 50\}$; $L_4 = \{45, 19, 43, 21, 18\}$?

   (b) Que permet de réaliser cet algorithme concernant la course cycliste ?

Laurae nous a d'ailleurs fait remarquer une petite coquille dans l'énoncé. Il faut bien évidemment lire c:=rand(1,50) et non c:=and(1,50) qui comble de malchance aurait une toute autre signification pour le programmeur averti.

Et Adriweb une deuxième: il manque un "ou" logique qui a sauté avec le retour à la ligne entre (b=d) et (b=e).

Vous ayant annoncé ce sujet dès sa disponibilité, il n'y a pas encore de corrigé en ligne même si cela ne saurait tarder. Ce n'est pas forcément gênant pour les exercices usuels, mais c'est plus embêtant pour les questions d'algorithmique.

Aussi, nous te proposons ci-dessous la première proposition de correction pour les questions d'algorithmique.

Attention: lire ces éléments de correction sans avoir au préalable lu, compris et cherché l'énoncé ne servira pas à grand chose...

Maths BAC S Inde Avril 2012 - élément de correction par Xavier Andréani

Exercice 1:

Partie A:

1)
L'on choisit ici un groupe de 5 coureurs parmi 50.
On ne peut pas avoir plusieurs fois le même coureur dans un même groupe, et les coureurs ne sont pas ordonnés au sein de ces groupes.
Il n'y a donc ni répétition ni ordre: c'est assimilable un un tirage simultané de 5 coureurs parmi 50, ce qui va se dénombrer avec une combinaison.
Le nombre de groupes différents réalisables est donc nCr(50,5)=50!/(5!*(50-5)!)=50!/(5!*45!)=2118760

Pack et mise en page par Laurae, rédigé par critor sur le forum, administrateurs de TI-Planet.org

2) a)
Les ensembles sont affichés sont la forme {a,b,c,d,e}.

Pour l'ensemble L1={2,11,44,2,15}, on remarque que a=d.
Or, il n'est pas possible de sortir de la boucle "tant que" avec la condition a=d qui au contraire est une condition suffisante pour continuer la boucle.
L'algorithme ici présenté ne peut donc pas produire L1.

De même on a pour l'ensemble L3={12,17,23,17,50} b=d, et l'algorithme ne peut donc pas produire L3.

Le seuls ensembles pouvant être produits par l'algorithme sont donc L2 et L4.

2) b)
L'algorithme est constitué d'une boucle "tant que".

Le corps de cette boucle tire aléatoirement 5 entiers a, b, c, d, e entre 1 et 50.

Pour savoir quelle tâche réalise la boucle "tant que", il faut connaître la condition d'arrêt de cette boucle.
Cette condition d'arrêt est la négation logique de la condition de répétition du tant que qui apparaît dans l'algorithme, ce qui donne en sortie de la boucle "tant que":
(a≠b) et (a≠c) et (a≠d) et (a≠e) et (b≠c) et (b≠d) et (b≠e) et (c≠d) et (c≠e) et (d≠e)
C'est-à-dire que les 5 entiers a, b, c, d, e doivent être tous distincts deux à deux.

Donc l'algorithme tire 5 entiers entre 1 et 50, jusqu'à-ce qu'ils soient tous distincts deux à deux.

Ce qui se reformule encore plus simplement: l'algorithme tire 5 entiers entre 1 et 50 distincts deux à deux.

N'hésite pas à venir demander de l'aide ou des conseils sur notre chat et notre forum.

Pack et mise en page par Laurae, rédigé par critor sur le forum, administrateurs de TI-Planet.org

Correction 2 : Eléments correction algo BAC S maths Amérique du Nord 2012

de **critor** » 01 Juin 2012 08:42

Dans une news précédente, nous te confirmions pour la 2ème fois en 3 sujets de maths inédits du BAC S 2012 que tu pouvais être interrogé dans ton épreuve sur les ajouts du nouveau programme de maths commencé en Seconde en 2009, et notamment l'algorithmique.

Comme nous l'avions déjà fait la dernière fois pour le sujet de maths tombé en Inde en avril dernier, nous allons te donner quelques éléments de correction sur la question d'algorithmique puisqu'il n'y en a aucune de ce type dans tes annales papier.

Voici donc ce qui est tombé hier en Amérique du Nord:

a) Montrer que, pour tout entier naturel k supérieur ou égal à 2, la distance M_kN_k entre les points M_k et N_k est donnée par $M_kN_k = \frac{\ln(k)}{k}$.

b) Écrire un algorithme déterminant le plus petit entier k_0 supérieur ou égal à 2 tel que la distance M_kN_k soit inférieure ou égale à 10^{-2}.

Remarque: Il s'agit ici de l'application de la convergence de ln(k)/k vers 0 (non démontrée ici), impliquant qu'à partir d'un certain rang, ln(k)/k devient irrémédiablement inférieure à toute valeur positive, ici 10^{-2}.
En conséquence, les deux courbes (C) et (D) sont asymptotes puisque leur écart tend vers 0.

Le nouveau programme de maths détaille trois compétences pour l'algorithmique:
- Savoir dérouler et interpréter un algorithme écrit en langage naturel
- Savoir produire un algorithme en langage naturel pour répondre à un problème
- Savoir implémenter un algorithme sur sa calculatrice programmable ou sur un logiciel de mathématiques

C'est ici la deuxième qui est évaluée.

Attention, petite confusion fréquente que nous avons souvent constatée dans les questions d'utilisateurs sur notre chat et notre forum. Il ne s'agit absolument pas de produire un **programme** pour calculatrices qui ont toutes des langages différents, mais un **algorithme**, c'est-à-dire une suite d'instructions rédigées dans un **langage naturel** supérieur à toutes les machines et compréhensible par tous sans formation particulière, appelé aussi **pseudo-code**.

L'on doit donc ici rechercher un entier sur un intervalle infini. Une boucle "pour" n'est donc pas appropriée et nous allons utiliser une boucle "tant que" avec comme condition de poursuite de la boucle le contraire logique de la condition recherchée. La boucle "tant que" s'arrête alors forcément sur la condition recherchée.

Ceci étant dit, l'on peut procéder par balayage exactement comme pour l'algorithme de réalisation d'un tableau de valeurs.

Voici sans plus attendre notre proposition de correction, respectant les conseils de rédaction des IREM:

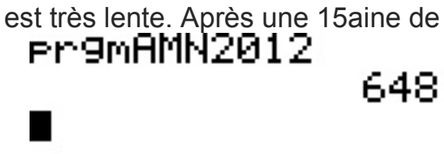
Algorithme: recherche_par_balayage
Entrées: aucune
Résultat: k, le plus petit entier supérieur ou égal à 2 tel que $\frac{\ln(k)}{k} \leq 10^{-2}$
Variables: k (nombre entier)
Début algorithme
    k←2;
    Tant que $\frac{\ln(k)}{k} > 10^{-2}$ faire
        k←k+1;
    Fin tant que;
    retourne k;
Fin algorithme.

Pour vérifier que notre algorithme est correct, on peut l'implémenter en un programme sur calculatrice, mais notez bien que ce n'est pas ça que vous devrez copier sur votre copie - ne confondez pas algorithme et programme.

Voici le programme équivalent pour TI-73 à TI-84:

```
PROGRAM:AMN2012
:2→K
:While ln(K)/K>1
0^-2
:K+1→K
:End
:K
```

L'exécution en est très longue, puisque la convergence vers 0 de ln(k)/k est très lente. Après une 15aine de secondes sur TI-84, on obtient le résultat:



Les TI-82 Stats et TI-83 Plus ayant un processeur cadencé à 6MHz au lieu de 15MHz comme les TI-84, il faudra patienter une 30aine de secondes.

N'hésite pas à venir demander de l'aide ou des conseils sur notre chat et notre forum et à revenir pour découvrir les questions d'algo des prochains sujets de maths du BAC S 2012 qui tomberont avant ton épreuve!

Pack et mise en page par Laurae, rédigé par critor sur le forum, administrateurs de TI-Planet.org

Correction 3 : Eléments correction algo BAC S maths Polynésie 2012

de **critor** » 09 Juin 2012 10:53

Dans une news précédente, nous te confirmions pour la 3ème fois en 4 sujets de maths inédits du BAC S 2012 que tu pouvais être interrogé dans ton épreuve sur les ajouts du nouveau programme de maths commencé en Seconde en 2009, et notamment l'algorithmique.

Comme nous l'avions déjà fait les 2 fois précédentes, nous allons te donner quelques éléments de correction sur la question d'algorithmique puisqu'il n'y en a aucune de ce type dans tes annales papier.

Voici donc ce qui est tombé hier en Polynésie française:

EXERCICE 3 (5 points)

Partie A

On considère l'algorithme suivant :

les variables sont le réel U et les entiers naturels k et N.

Entrée
Saisir le nombre entier naturel non nul N

Traitement
Affecter à U la valeur 0
Pour k allant de 0 à $N-1$
  Affecter à U la valeur $3U-2k+3$
Fin pour
Sortie
Afficher U

Quel est l'affichage en sortie lorsque N = 3 ?

Il s'agit donc de dérouler l'algorithme tout en conservant l'état de l'ensemble des variables à chaque instruction, pour donner la valeur finale de la variable U.

Une méthode type que tu aurais du apprendre en Seconde dès 2009 est de construire un tableau que l'on va appeler "trace de l'algorithme", avec une colonne pour chaque variable utilisée. Il suffit ensuite de dérouler l'algorithme et d'ajouter une ligne au tableau à chaque modification d'une valeur de variable.

Le voici avec même des commentaires en prime:

N	U	k	commentaires
3	non initialisé	non initialisé	Saisie de N=3
3	**0**	non initialisé	U reçoit 0
3	0	**0**	début de la boucle « pour » avec k qui reçoit 0
3	**3**	0	U reçoit 3U-2k+3
3	3	**1**	2ème passage dans la boucle « pour » avec k qui reçoit 1
3	**10**	1	U reçoit 3U-2k+3
3	10	**2**	3ème et dernier passage dans la boucle « pour » avec k qui reçoit 2=N-1
3	**29**	2	U reçoit 3U-2k+3

A la dernière ligne, on obtient donc U=29.

Remarquons que cet algorithme très simple est facilement transposable en un programme pour calculatrice graphique. Le voici traduit en TI-Basic pour toutes les calculatrices TI-73 à TI-86:

```
PROGRAM:POLY2012
:Input N
:0→U
:For(K,0,N-1
:3U-2K+3→U
:End
:U
```

Une simple exécution du programme nous donne alors instantanément la réponse qu'il suffirait de recopier!

```
Pr9mPOLY2012
?3
              29
```

Voici la même chose pour TI-Nspire:

Mais hélas, ce programme ne nous donne pas la justification, c'est-à-dire la trace de l'algorithme... Et sauf mention contraire dans l'énoncé, tout résultat non justifié est pénalisé dans la notation...

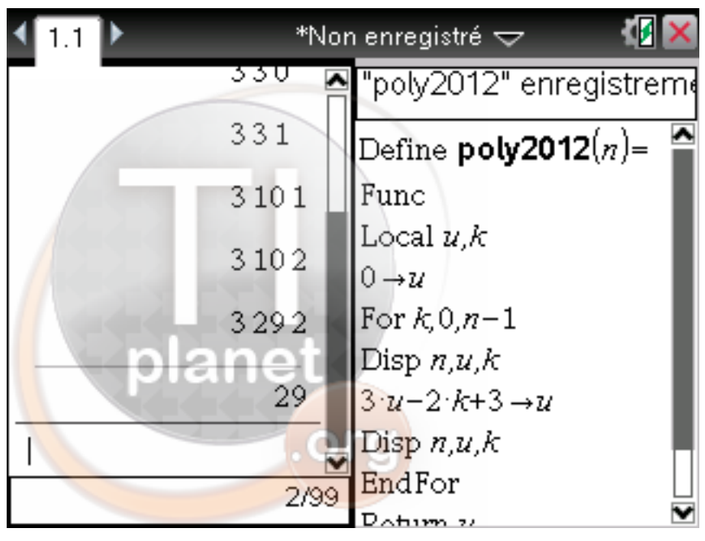
Et bien ce n'est pas grave! Modifions tout simplement le programme ci-dessus pour afficher dans l'ordre l'état des 3 variables N, U et k à chaque début et fin d'itération de la boucle "pour":

```
PROGRAM:POLY2012
:0→U
:For(K,0,N-1
:Disp {N,U,K
:3U-2K+3→U
:Disp {N,U,K
:End
:U
```

Nous obtenons alors la trace de l'algorithme qu'il suffit tout simplement de recopier!

```
 3 0 0
 3 0 1
 3 1 1
 3 1 0 1
 3 1 0 2
 3 2 9 2
 3 2 9 2
 3 29 29
```

La même chose est là encore réalisable sur TI-Nspire:



Et pour ceux qui ont des TI-89 à TI-Voyage 200, le code est quasiment identique:

```
:poly2012(n)
:Func
:Local u,k:0→u
:For k,0,n-1
:Disp n,u,k
:3*u-2*k+3→u
:Disp n,u,k
:EndFor
:Return u
```

(merci Loulou54)

Et si tu as une Casio, et bien c'est par ici -> http://www.planet-casio.com/Fr/forums/t ... trice.html

Et c'est là que l'on se dit "merci TI",
pour ce superbe outil !

Pack et mise en page par Laurae, rédigé par critor sur le forum, administrateurs de TI-Planet.org

Correction 4 : Eléments correction question calto BAC S Polynésie 2012

de **critor** » 10 Juin 2012 12:54

Hier, nous vous avions corrigé la partie A de l'exercice n°3 du sujet de maths du BAC S tombé vendredi en Polynésie française, et portant sur l'algorithmique.

Mais ce n'est pas la seule question "intéressante" de cet exercice... Voyons la suite:

Partie B

On considère la suite (u_n) définie par $u_0 = 0$ et, pour tout entier naturel n, $u_{n+1} = 3u_n - 2n + 3$.

1) Calculer u_1 et u_2.

2) a) Démontrer par récurrence que, pour tout entier naturel n, $u_n \geqslant n$.

b) En déduire la limite de la suite (u_n).

3) Démontrer que la suite (u_n) est croissante.

4) Soit la suite (v_n) définie, pour tout entier naturel n, par $v_n = u_n - n + 1$.

a) Démontrer que la suite (v_n) est une suite géométrique.

b) En déduire que, pour tout entier naturel n, $u_n = 3^n + n - 1$.

5) Soit p un entier naturel non nul.

a) Pourquoi peut-on affirmer qu'il existe au moins un entier n_0 tel que, pour tout $n \geqslant n_0$, $u_n \geqslant 10^p$?

On s'intéresse maintenant au plus petit entier n_0.

b) Justifier que $n_0 \leqslant 3p$.

c) Déterminer à l'aide de la calculatrice cet entier n_0 pour la valeur $p = 3$.

La question B) 5) c) demande ici de trouver un résultat à la calculatrice.

Réponse qui ne pourra donc pas être justifiée puisque l'usage d'imprimantes est interdit pendant l'épreuve.
Au mieux, vous ne pourrez que raconter ce que vous faites.

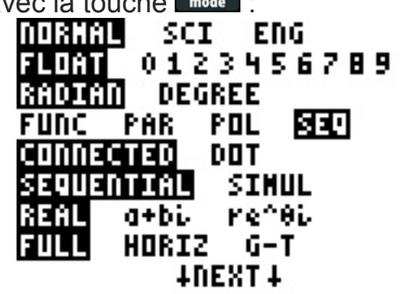
Habituellement, ce type de question tombe plutôt pour déterminer une valeur approchée ou un encadrement juste après une utilisation du théorème de la bijection dit "des valeurs intermédiaires", ou alors en série ES.

Nous sommes très étonnés de trouver une telle question cette année en série S, mais elle n'en est pas moins intéressante.

Nous allons surtout vous montrer que nous pouvons traiter cette question sans avoir rien fait/compris dans les questions précédentes, et que si vous savez gérer votre stress et garder les yeux en face des trous vous pouvez très facilement gagner des points sans effort car:
- beaucoup de questions donnent les résultats qu'elles vous demandent de démontrer (questions du type "montrez que...")
- il y a un enchaînement logique dans l'énoncé, et pour répondre à une question il suffit presque toujours d'utiliser la réponse de la question ou partie précédente (et donc si vous êtes bloqués au BAC, astuce très efficace: relisez la question ou partie précédente)

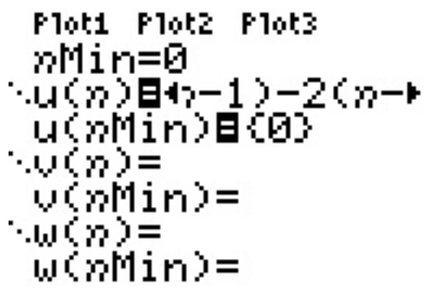
Récapitulons ici les choses
- On nous demande donc ici à la question B)5)c) de trouver un entier n_0 tel que pour tout $n \geq n_0$, $u_n \geq 10^p$ avec p=3, soit ici $u_n \geq 1000$. En langage naturel, on cherche donc le rang n_0 à partir duquel la suite u_n dépasse irrémédiablement 1000.
- D'après la question précédente B)5)b), on sait de plus que $n_0 \leq 3p$, soit $0 \leq n_0 \leq 9$, ce qui réduit fort avantageusement l'espace de recherche.
- La question B)3) nous dit que la suite est croissante, et donc la plus petite valeur n_0 trouvée telle que $u_{n0} \geq 1000$ conviendra.
- Et la question B)5)a) a gentillesse de nous garantir que de tels n_0 existent!

Et bien allons-y: passons notre calculatrice TI-73 à TI-84 en mode suite avec la touche quitter QUIT mode :

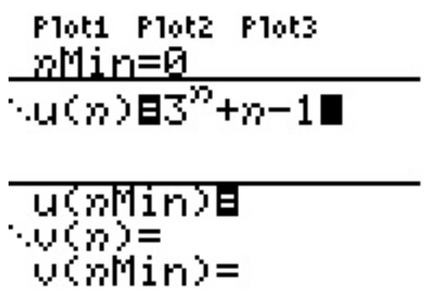


Pack et mise en page par Laurae, rédigé par critor sur le forum, administrateurs de TI-Planet.org

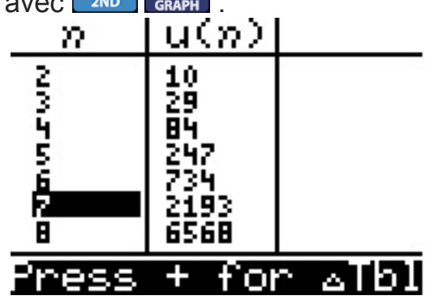
Accédons ensuite à l'éditeur de suites avec la touche [graph stats STAT PLOT f1 f(x) Y=] pour entrer la suite récurrente définie par $u_0=0$ et $u_{n+1}=3u_n-2n+3$.
Notons que la calculatrice ne permet pas de définir u_{n+1} mais uniquement u_n. Pour entrer une telle suite, il faut donc remplacer toutes les occurrences de la variable n par des (n-1), et rentrer $u_n=3u_{n-1}-2(n-1)+3$.



Notons qu'il y a ici beaucoup plus simple si l'on n'a pas oublié la question B)4)b) qui donne la formule générale $u_n=3^n+n-1$, formule beaucoup plus facile à saisir sans erreur et de plus fiable puisque donnée par l'énoncé!



Une fois notre suite saisie, demandons simplement un tableau de valeurs avec [2nde 2ND] [table graphe GRAPH] :



n	u(n)
2	10
3	29
4	84
5	247
6	734
7	2193
8	6568

Press + for ▲Tbl

Et voilà, on trouve en effet quasi immédiatement que la suite u_n dépasse irrémédiablement 1000 à partir de $n_0=7$.

Pack et mise en page par Laurae, rédigé par critor sur le forum, administrateurs de TI-Planet.org

Correction 5 : Eléments correction 2ème algo BAC S maths Polynésie 2012

de **critor** » 10 Juin 2012 14:49

Dans une news précédente, nous te confirmions avec le sujet de Polynésie française, pour la 3ème fois en 4 sujets de maths inédits du BAC S 2012, que tu pouvais être interrogé dans ton épreuve sur les ajouts du nouveau programme de maths commencé en Seconde en 2009, et notamment l'algorithmique.

Comme nous l'avions déjà fait les 2 fois précédentes, nous te donnions alors quelques éléments de corrections sur l'algorithme qu'il fallait analyser en partie A de l'exercice 3.

Le nouveau programme de maths détaillait trois compétences évaluables pour l'algorithmique:
- Savoir dérouler et interpréter un algorithme écrit en langage naturel
- Savoir produire un algorithme en langage naturel pour répondre à un problème
- Savoir implémenter un algorithme sur sa calculatrice programmable ou sur un logiciel de mathématiques

La partie A concernait donc la 1ère compétence.

Mais si nous allons un peu plus loin, la dernière question de la partie B du même exercice demande de produire un algorithme!

Cela fait donc deux questions d'algorithmique dans le même sujet!

Voyons donc un peu ça aujourd'hui:

EXERCICE 3 (5 points)

Partie A

On considère l'algorithme suivant :

les variables sont le réel U et les entiers naturels k et N.

Entrée
Saisir le nombre entier naturel non nul N

Traitement
Affecter à U la valeur 0
Pour k allant de 0 à $N-1$

  Affecter à U la valeur $3U-2k+3$

Fin pour
Sortie
Afficher U

Quel est l'affichage en sortie lorsque N = 3 ?

Partie B

On considère la suite (u_n) définie par $u_0 = 0$ et, pour tout entier naturel n, $u_{n+1} = 3u_n - 2n + 3$.

1) Calculer u_1 et u_2.

2) a) Démontrer par récurrence que, pour tout entier naturel n, $u_n \geqslant n$.

   b) En déduire la limite de la suite (u_n).

3) Démontrer que la suite (u_n) est croissante.

4) Soit la suite (v_n) définie, pour tout entier naturel n, par $v_n = u_n - n + 1$.

   a) Démontrer que la suite (v_n) est une suite géométrique.

   b) En déduire que, pour tout entier naturel n, $u_n = 3^n + n - 1$.

5) Soit p un entier naturel non nul.

   a) Pourquoi peut-on affirmer qu'il existe au moins un entier n_0 tel que, pour tout $n \geqslant n_0$, $u_n \geqslant 10^p$?

   On s'intéresse maintenant au plus petit entier n_0.

   b) Justifier que $n_0 \leqslant 3p$.

   c) Déterminer à l'aide de la calculatrice cet entier n_0 pour la valeur $p = 3$.

   d) Proposer un algorithme qui, pour une valeur de p donnée en entrée, affiche en sortie la valeur du plus petit entier n_0 tel que, pour tout $n \geqslant n_0$, on ait $u_n \geqslant 10^p$.

Il s'agit donc de produire un algorithme de recherche du rang à partir duquel la suite dépasse une certaine valeur, ici $u_n \geq 1000$.

C'est donc exactement le même type d'algorithme que celui demandé dans le sujet d'Amérique du Nord.

Nous allons donc produire un algorithme similaire de recherche par balayage en incrémentant le rang k à partir de 0.
Pour cela, nous utiliserons une boucle "tant que" avec comme condition de poursuite le contraire logique de la condition recherchée, ici: $u_k < 1000$.

Pour le calcul de la valeur de u_n dans la condition, utilisons tout simplement la formule générale de la suite donnée à la question B)4)b): $u_n = 3^n + n - 1$.

Algorithme: recherche_par_balayage
Entrées: p *(nombre entier naturel non nul)*
Résultat: le plus petit entier n_0 tel que $\forall n \geq n_0, u_n \geq 10^p$
Variables: k *(nombre entier)*
Début algorithme
    k←0;
    Tant que $3^k + k - 1 < 10^p$ faire
        k←k+1;
    Fin tant que
    retourne k;
Fin algorithme

Afin de vérifier si notre algorithme semble correct, on peut le traduire en un programme sur calculatrice TI-73 à TI-84:

```
PROGRAM:POLY12B
:Input P
:0→K
:While 3^K+K-1<1
0^P
:K+1→K
:End
:K
```

Contrairement au programme d'Amérique du Nord, nous obtenons cette fois-ci un résultat en un temps raisonnable!

```
prgmPOLY12B

?3
                7
```

Le résultat 7 confirme d'ailleurs celui trouvé à la question précédente dans la news précédente!

Notons qu'il était également possible d'utiliser la définition par récurrence de la suite u_n pour calculer la condition. Cela nécessite d'utiliser 2 variables: une pour stocker le rang de la suite (k) et une 2ème pour stocker la valeur précédente de la suite nécessaire au calcul par récurrence.

Cette récurrence est donc un petit peu plus compliqué du point de vue algorithmique, mais très similaire à ce que fait l'algorithme de la partie A qui n'est donc pas là pour rien.

Algorithme: recherche_par_balayage
Entrées: p *(nombre entier naturel non nul)*
Résultat: le plus petit entier n_0 tel que $\forall n \geq n_0, u_n \geq 10^p$
Variables: k *(nombre entier)*, u *(nombre entier)*
Début algorithme
    k←0;
    u←0;
    Tant que $u < 10^p$ faire
        u←3u-2k+3;
        k←k+1;
    Fin tant que
    retourne k;
Fin algorithme

Là encore, après modification du programme, la calculatrice confirme le résultat précédent:

```
PROGRAM:POLY12B
:0→K
:0→U
:While U<10^P
:3U-2K+3→U
:K+1→K
:End
:K
```

Correction 6 : Eléments correction algo BAC S maths Centres Etrangers 2012

de **critor** » 14 Juin 2012 00:55

Dans la news précédente, nous te publiions le 5ème sujet de maths du BAC S 2012 qui venait de tomber dans les lycées français des Centres Etrangers (Europe et Afrique).

Et pour la 4ème fois en 5 sujets, tu as eu droit à de l'algorithmique en exercice 2. Comme nous l'avons déjà fait dans plusieurs news précédentes, voyons un peu ça de plus près:

Exercice 2 (5 points)

Commun à tous les candidats

On considère la suite (I_n) définie pour n entier naturel non nul par :

$$I_n = \int_0^1 x^n e^{x^2} dx.$$

1. a) Soit g la fonction définie sur **R** par $g(x) = x e^{x^2}$.

Démontrer que la fonction G définie sur **R** par $G(x) = \frac{1}{2} e^{x^2}$ est une primitive sur **R** de la fonction g.

b) En déduire la valeur de I_1.

c) À l'aide d'une intégration par parties, démontrer que, pour tout entier n, supérieur ou égal à 1, on a :

$$I_{n+2} = \frac{1}{2} e - \frac{n+1}{2} I_n.$$

d) Calculer I_3 et I_5.

2. On considère l'algorithme suivant :

Initialisation	Affecter à n la valeur 1 Affecter à u la valeur $\frac{1}{2}e - \frac{1}{2}$
Traitement	Tant que $n < 21$ Affecter à u la valeur $\frac{1}{2}e - \frac{n+1}{2}u$ Affecter à n la valeur $n+2$
Sortie	Afficher u

Quel terme de la suite (I_n) obtient-on en sortie de cet algorithme ?

Cette fois-ci donc, pas question de produire un algorithme. L'algorithme est donné et il s'agit de l'interpréter.

Mais la question nous mâche le travail: l'algorithme calcule la valeur d'un terme de la suite (I_n) définie en introduction à l'aide d'une intégrale.

Comme il n'y a aucune intégrale dans cet algorithme, qu'il y a des "n+2", et que de plus l'avant dernière affectation ressemble énormément à la formule de récurrence d'ordre 2 donnée au 1)c), on peut penser que l'algorithme effectue le calcul par récurrence.

En initialisation, la variable "n" est affecté avec la valeur 1, rang initial de la suite (I_n).
La variable "u" reçoit quant à elle la valeur e/2-1/2, qui est la valeur de I_1 que vous avez normalement trouvée au 1)b). (et oui - au BAC, les données parachutées en cours d'exercices sont très souvent des réponses des questions précédentes)

Les variables "n" et "u" sont toujours modifiées ensembles par la boucle "tant que".

Lorsque l'on sort de la boucle, c'est que la condition n<21 de poursuite du "tant que" vient de se révéler fausse. La variable "n" vaut alors 21.
Et il serait logique de supposer sans chercher à comprendre plus en détail l'algorithme que la variable "u" contient alors la valeur de I_{21}.

La réponse serait donc 21.

Mais peut-être que cela vous paraît trop simple? Peut-être qu'il y aurait un piège?
Vérifions à la calculatrice. Traduisons cet algorithme en un programme pour TI-73 à TI-84:

```
PROGRAM:CETR2012
:1→N
:e/2-1/2→U
:While N<21
:e/2-(N+1)/2U→U
:N+2→N
:End
:U
```

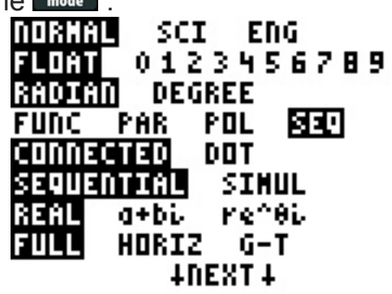
L'exécution de ce programme nous donne la valeur du terme de la suite (I_n) calculé.

```
Prgm CETR2012
     .1140007275
```

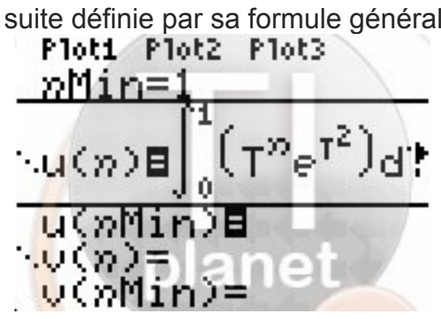
Mais ne tombons pas dans le piège: ce n'est pas la valeur de ce terme qui nous est demandée mais son rang, et l'algorithme ne répond pas à la question: ce n'est qu'une étape.

Pack et mise en page par Laurae, rédigé par critor sur le forum, administrateurs de TI-Planet.org

Passons donc notre calculatrice TI-73 à TI-84 en mode suite avec la touche quitter QUIT mode :



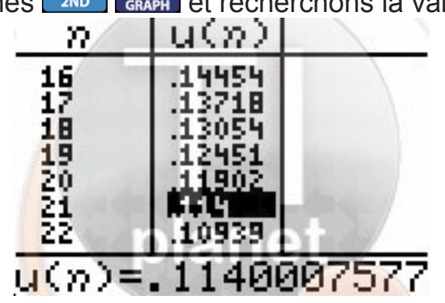
Accédons ensuite à l'éditeur de suites avec la touche graph stats STAT PLOT F1 f(x) Y= pour entrer la suite définie par sa formule générale avec l'intégrale:



Notez qu'ici j'ai du remplacer toutes les variables "X" de la formule par des variables "T". En effet, ma TI-84 munie de l'OS 2.55MP me harcèle obstinément avec des erreurs "Invalide" si j'utilise une seule variable "X" dans une formule de suite, pensant sans doute que je me trompe et confonds avec une saisie de fonction. Si l'intention est louable (empêcher les utilisateurs de se tromper), à partir du moment où cela empêche de saisir des expressions mathématiquement correctes, je considère cela comme un bug.

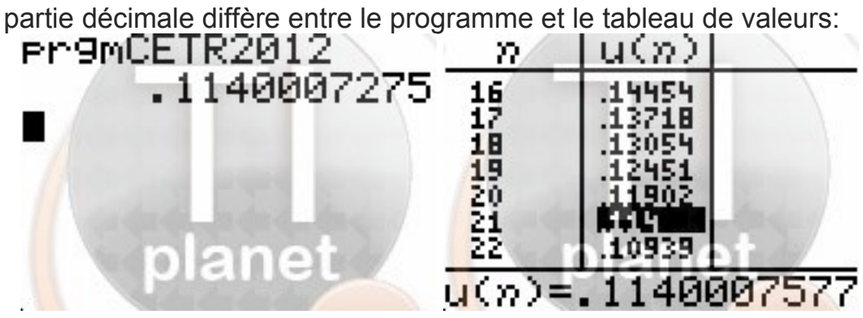
Autre petite remarque. Il n'était pas possible ici directement de définir la suite par récurrence sur TI-73 à TI-84 qui nous auraient là encore invectivés avec des erreurs "Invalide". Il s'agit en effet d'une récurrence d'ordre 2, non supportée par la calculatrice. Il existe toutefois un moyen de passer outre en définissant un système de deux suites récurrentes d'ordre 1. Mais peu importe ici puisque l'on avait la formule générale.

Une fois la suite définie, demandons son tableau de valeur avec les touches 2nde 2ND table graphe GRAPH et recherchons la valeur renvoyée par le programme ci-dessus:



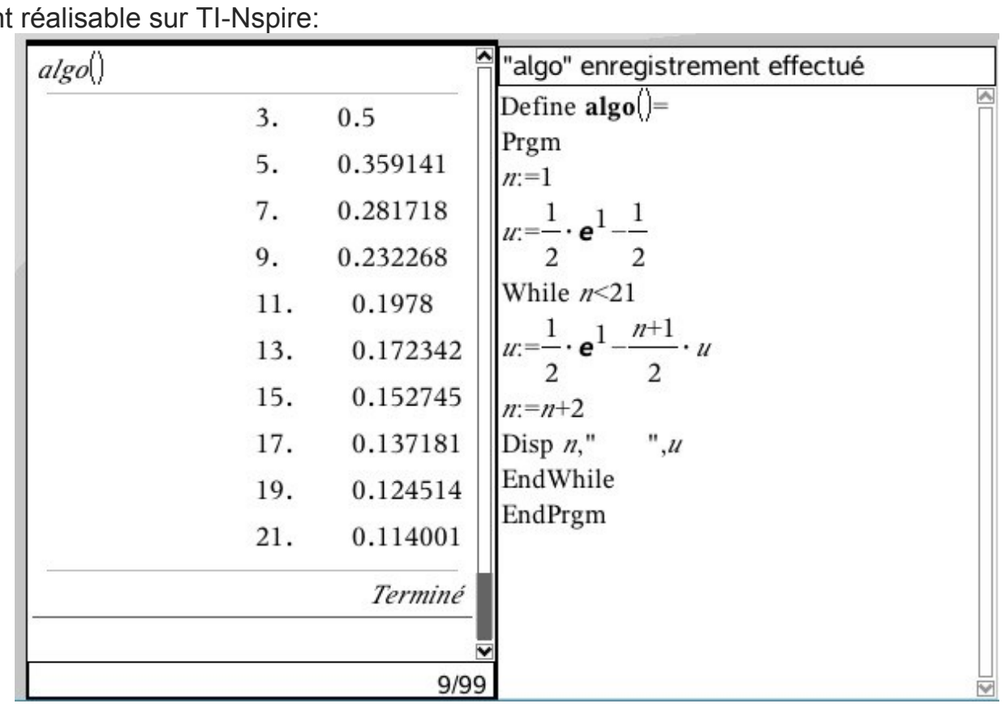
Et la valeur correspond bien au terme de rang 21 - il n'y avait donc pas de piège.

Autre petite remarque: vous constaterez que la fin de la partie décimale diffère entre le programme et le tableau de valeurs:



La calculatrice ne travaille pas sur l'ensemble des réels mais sur des nombres décimaux de 13 chiffres multipliés par une puissance de 10.
Chaque calcul sur un nombre non décimal est donc l'objet d'une petite erreur d'approximation.
Et bien la différence vient du fait que le programme obtient le résultat par récurrence, c'est-à-dire en effectuant plusieurs calculs, et par conséquent plusieurs approximations qui s' "additionnent".

Le même style de programme est bien évidemment réalisable sur TI-Nspire:



A bientôt!

Pack et mise en page par Laurae, rédigé par critor sur le forum, administrateurs de TI-Planet.org

Correction 7 : Eléments correction Algo Obligatoire BAC S 2012 Antilles

de **critor** » 19 Juin 2012 17:40

Pour la 5ème fois en 6 sujets de Maths du BAC S 2012, l'algorithmique a encore frappé avec le dernier sujet tombé aux Antilles-Guyane-Guadeloupe-Martinique.

Pour les candidats ayant l'épreuve obligatoire, contrairement aux derniers sujets précédents, le contexte n'est pas les suites mais à nouveau les probas comme en Inde au mois d'avril.

Le voici:

5. On considère l'algorithme :

```
A et C sont des entiers naturels,
C prend la valeur 0
Répéter 9 fois
    A prend une valeur aléatoire entière entre 1 et 7.
        Si A > 5 alors C prend la valeur de C + 1
        Fin Si
Fin répéter
Afficher C.
```

Dans l'expérience aléatoire simulée par l'algorithme précédent, on appelle X la variable aléatoire prenant la valeur C affichée.
Quelle loi suit la variable X ? Préciser ses paramètres.

Il s'agit donc d'un algorithme simulant une expérience aléatoire et retournant des valeurs considérées comme celle d'une variable aléatoire. On demande de décrire la loi de probabilité suivie par cette variable.

Avec le nouveau programme qui a supprimé certaines lois (loi exponentielle, loi continue à densité), le choix est désormais assez restreint:
- loi discrète quelconque à décrire par un tableau
- loi de Bernoulli / loi binomiale
- loi continue uniforme

Que fait donc cet algorithme?

Il tire 9 fois un nombre aléatoire 'A' entre 1 et 7.
Il y a donc 9 répétitions d'une même expérience identique, et on s'orienterait vers une loi binomiale avec comme premier paramère n=9.
La variable 'C' initialisée à 0 est incrémentée de 1 à chaque fois que la variable 'A' obtient 6 ou 7.
Elle agit donc comme un compteur, et compte le nombre de fois où l'on a tiré les nombres 6 ou 7, événement de probabilité 2/7.

Il s'agit donc ici d'une loi binomiale de paramètres n=9 et p=2/7.

A bientôt!

Pack et mise en page par Laurae, rédigé par critor sur le forum, administrateurs de TI-Planet.org

Correction 8 : Eléments correction Algo Spécialité BAC S 2012 Antilles

de **critor** » 19 Juin 2012 18:50

Pour la 5ème fois en 6 sujets de Maths du BAC S 2012, l'algorithmique est donc encore tombée, aujourd'hui aux Antilles-Guyane-Guadeloupe-Martinique.
Mais cette fois-ci, il y a même un algo pour en prime les candidats ayant choisi la spécialité!

Voyons donc un peu le tout premier algo de spécialité:

4. On considère l'algorithme suivant où $Ent\left(\frac{A}{N}\right)$ désigne la partie entière de $\frac{A}{N}$:

A et N sont des entiers naturels

Saisir A

N prend la valeur 1

Tant que $N \leq \sqrt{A}$

Si $\frac{A}{N} - Ent\left(\frac{A}{N}\right) = 0$ alors Afficher N et $\frac{A}{N}$

Fin si

N prend la valeur N+1

Fin Tant que.

Quels résultats affiche cet algorithme pour $A = 12$?
Que donne cet algorithme dans le cas général ?

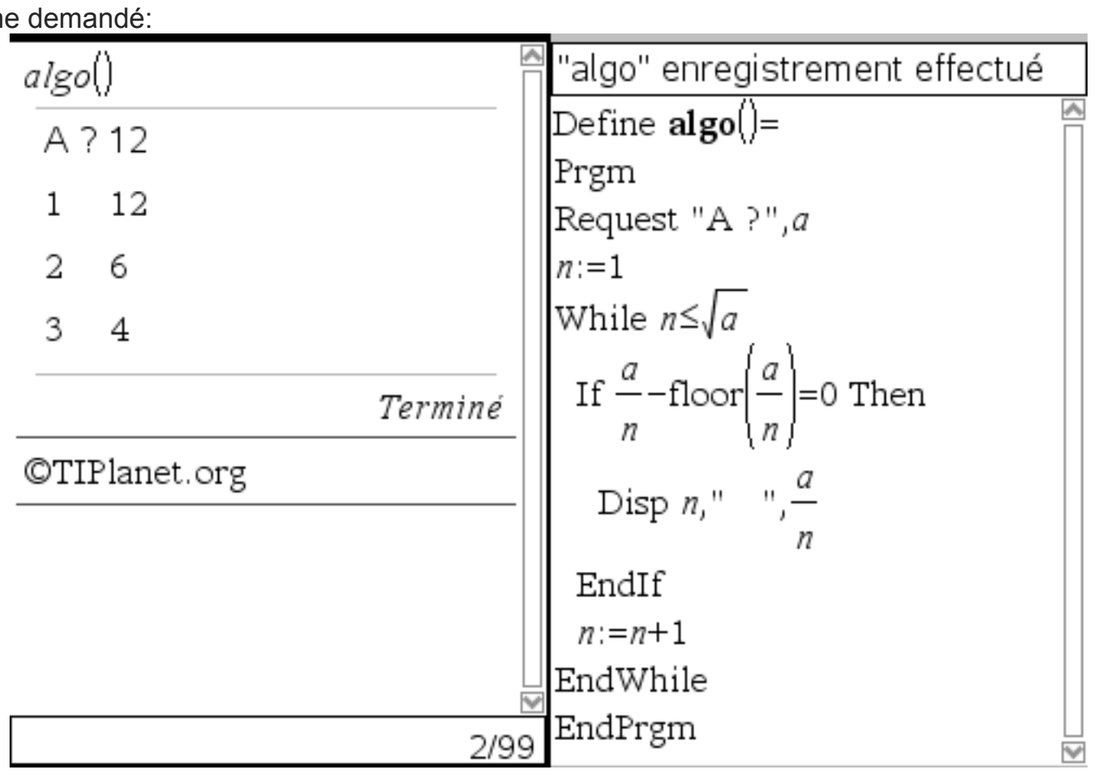
Ce n'est pas forcément bien compliqué, mais ce n'est clairement pas du même acabit que tout ce que nous avons vu jusqu'à présent.

Lorsque A/N-Ent(A/N)=0, cela veut dire que A/N=Ent(A/N).
Le nombre A/N est alors égal à sa partie entière, ce qui veut dire que A/N est un nombre entier, et que A est divisible par N.
Les nombres N et A/N affichés dans ce cas sont alors deux diviseurs de A, avec N plus petit que A/N.

Le fait de s'arrêter à la racine de A évite de poursuivre inutilement la recherche: on obtiendrait les mêmes paires de diviseurs avec avec N plus grand que A/N.

On peut donc supposer que ce programme recherche et affiche tous les diviseurs du nombre A.

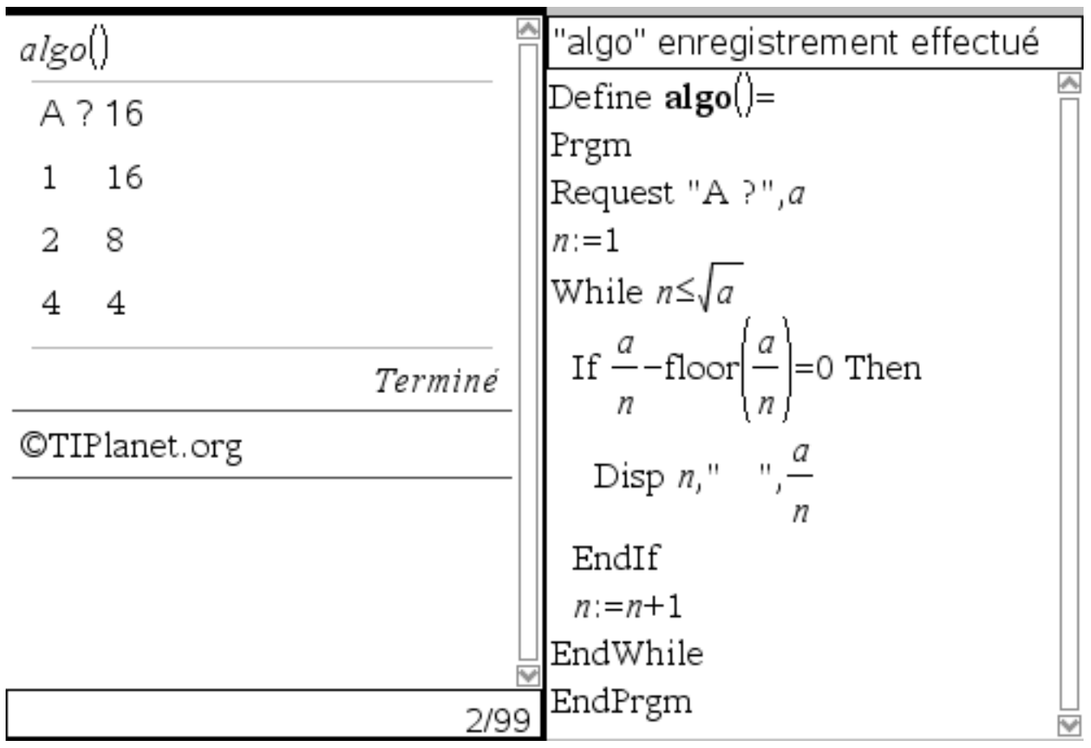
Testons un peu sur TI-Nspire avec A=12 comme demandé:



En effet, 12 admet 6 diviseurs ici affichés par paires: 1, 2, 3, 4, 6, 12.

Remarquons une toute petite négligence (ou bug). L'algorithme trouve bien tous les diviseurs, mais peut dans certains cas afficher le même plusieurs fois.

Prenons A=16:

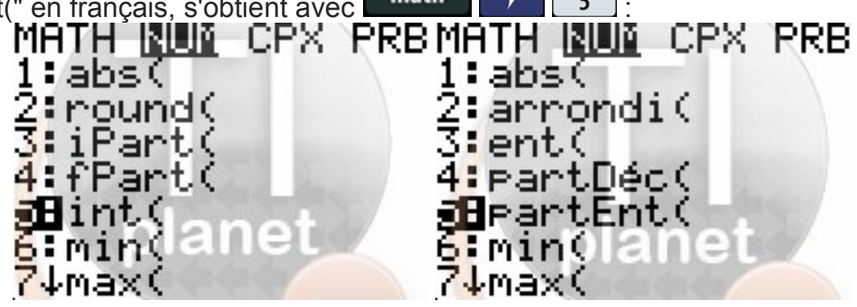


Il y a 5 diviseurs: 1, 2, 4, 8, 16, mais le 4 est affiché 2 fois.

La même chose est bien entendu réalisable sur les calculatrices TI-73 à TI-84.

Voici le listing pour celles qui fonctionnent en anglais et en français:

La fonction de partie entière, "int(" en anglais ou "partEnt(" en français, s'obtient avec :



La sortie du programme est bien évidemment la même:

A bientôt!

Pack et mise en page par Laurae, rédigé par critor sur le forum, administrateurs de TI-Planet.org

Correction 9 : Eléments correction algo Maths BAC S 2012 Asie

de **critor** » 20 Juin 2012 13:48

Pour la 6ème fois en 7 sujets, l'algo a encore frappé au BAC S!

Voici l'algo tombé dans les lycées français d'Asie ce matin:

Exercice 4 (5 points)

Commun à tous les candidats

1. On considère l'algorithme suivant :

Entrée	Saisir un réel strictement positif non nul a Saisir un réel strictement positif non nul b ($b > a$) Saisir un entier naturel non nul N
Initialisation	Affecter à u la valeur a Affecter à v la valeur b Affecter à n la valeur 0
Traitement	TANT QUE $n < N$ Affecter à n la valeur $n+1$ Affecter à u la valeur $\frac{a+b}{2}$ Affecter à v la valeur $\sqrt{\frac{a^2+b^2}{2}}$ Affecter à a la valeur u Affecter à b la valeur v
Sortie	Afficher u, afficher v

Reproduire et compléter le tableau suivant, en faisant fonctionner cet algorithme pour $a=4$, $b=9$ et $N=2$. Les valeurs successives de u et v seront arrondies au millième.

n	a	b	u	v
0	4	9		
1				
2				

On revient donc sur le combo algo/suites fort apprécié par les concepteurs de sujets: 4 exos d'algo sur 6 à ce jour.

L'algorithme fait en fait travailler sur le système de suites récurrentes d'ordre 1 qui va être étudié dans la suite de l'exercice.

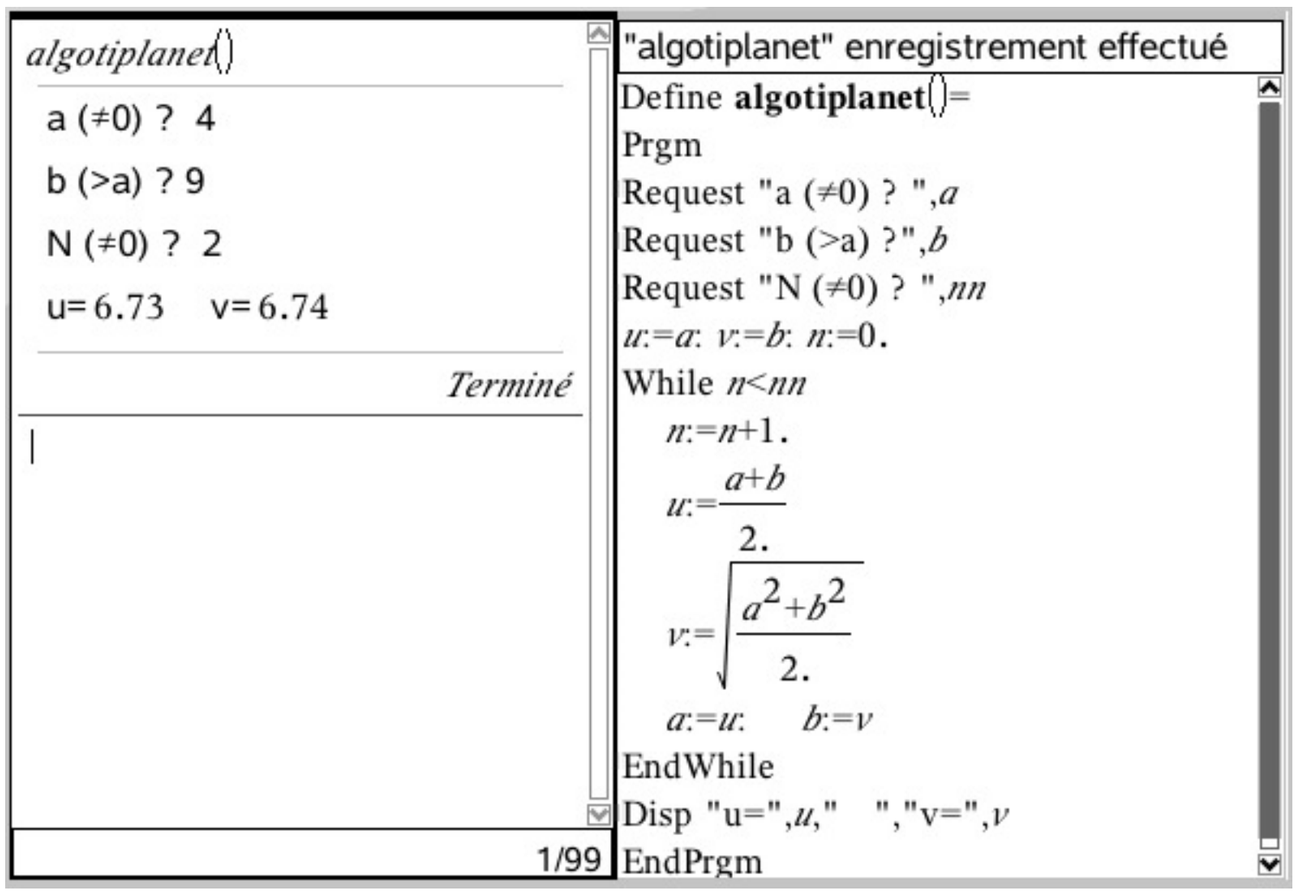
En fait il s'agit ici de réaliser la trace de l'algorithme, c'est-à-dire un tableau donnant l'état des variables à chaque itération de la boucle "tant que".
On peut bien évidemment dérouler l'algorihme de tête et compléter le tableau au fur et à mesure comme on avait déjà fait.

n	a	b	u	v	Commentaire
0	4	9	4	9	Fin de l'initialisation, juste avant le début de la boucle "tant que"
1	6.5	6.964	6.5	6.964	Fin de la 1ère itération de la boucle
2	6.732	6.736	6.732	6.736	Fin de la 2ème et dernière itération de la boucle

L'on peut traduire l'algorithme brut sous la forme d'un programme TI-Nspire.

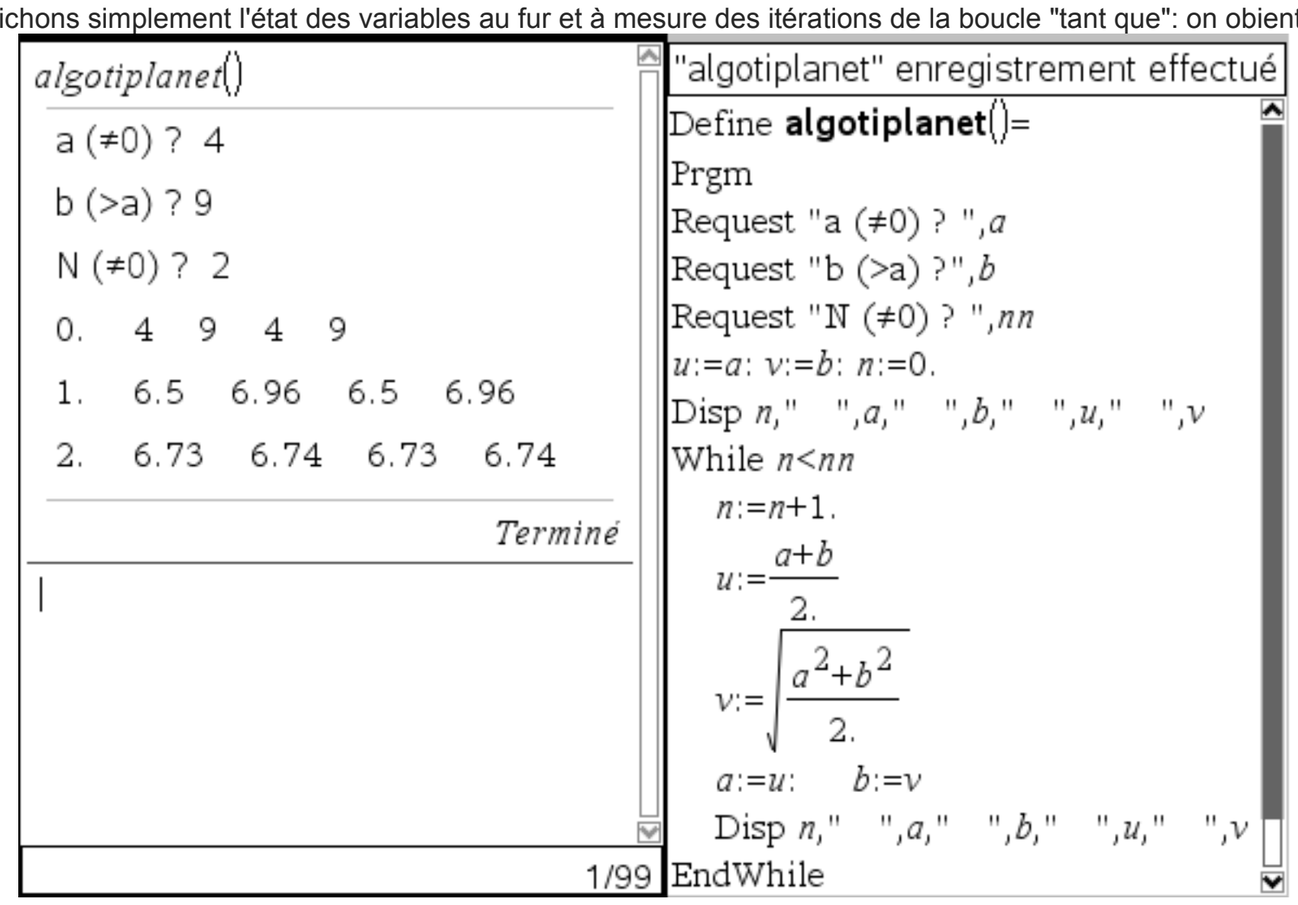
Pack et mise en page par Laurae, rédigé par critor sur le forum, administrateurs de TI-Planet.org

La Nspire ne différenciant pas variables minuscules et majuscules, l'on renomme ici la variable 'N' en 'nn'.



Mais on obtient simplement l'affichage des valeurs finales des variables u et v.

Et bien utilisons une super-astuce - affichons simplement l'état des variables au fur et à mesure des itérations de la boucle "tant que": on obient directement le tableau!

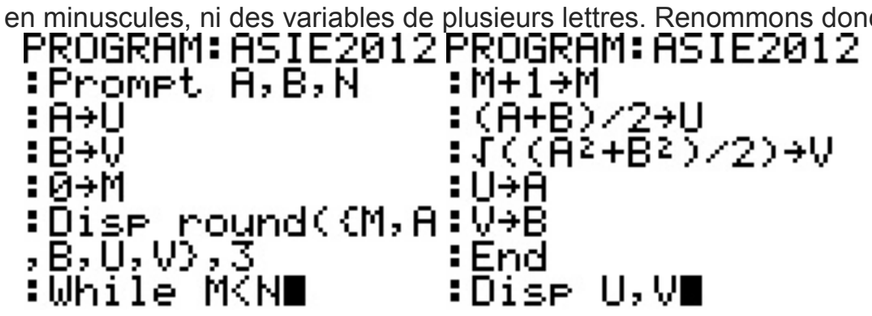


Notons que l'énoncé nous semble ambigu: on ne sait pas exactement à quoi correspondent les lignes du tableau qui sont au nombre de 3 alors qu'il n'y a que 2 itérations. Selon les endroits où on insère les instructions d'affichage, on obtiendra donc des tableaux différents, mais qui conduiront aux mêmes résultats.

Il me semblerait logique qu'une ligne du tableau fasse suite à 5 affectations, mais comme vous le voyez dans ce cas, les couples de variables a u et b v ont des valeurs identiques.

On peut bien évidemment réaliser la même chose sur TI-73 à TI-84

Là, la calculatrice ne permet pas d'utiliser des variables en minuscules, ni des variables de plusieurs lettres. Renommons donc la variable 'n' en 'M'.



On obtient bien évidemment le résultat final:

```
A=?4
B=?9
N=?2
      6.732097069
      6.736096793
             Done
```

Pour afficher des résultats à 3 chiffres après la virgule, on peut utiliser la fonction "arrondi(" ou "round(" selon que la calculatrice travaille en français ou en anglais:

```
MATH NUM CPX PRB  MATH NUM CPX PRB
1:abs(            1:abs(
2:round(          2:arrondi(
3:iPart(          3:ent(
4:fPart(          4:partDéc(
5:int(            5:partEnt(
6:min(            6:min(
7↓max(            7↓max(
```

Voici donc le listing du programme en mode anglais et français, utilisant la même astuce que ci-dessus:

```
PROGRAM:ASIE2012  PROGRAM:ASIE2012
:Prompt A,B,N     :Prompt A,B,N
:A→U              :A→U
:B→V              :B→V
:0→M              :0→M
:Disp round(M,A   :Disp arrondi(M
,B,U,V),3         ,A,B,U,V),3
:While M<N        :While M<N
PROGRAM:ASIE2012  PROGRAM:ASIE2012
:While M<N        :While M<N
:M+1→M            :M+1→M
:(A+B)/2→U        :(A+B)/2→U
:√((A²+B²)/2)→V   :√((A²+B²)/2)→V
:U→A              :U→A
:V→B              :V→B
:Pause round(M    :Pause arrondi(
PROGRAM:ASIE2012  PROGRAM:ASIE2012
:√((A²+B²)/2)→V   :√((A²+B²)/2)→V
:U→A              :U→A
:V→B              :V→B
:Pause arrondi(M  :Pause round(M,
M,A,B,U,V),3      A,B,U,V),3
:End              :End
:Disp U,V         :Disp U,V
```

Et sans surprise, le même résultat:

```
N=?2                N=?2
      {0 4 9 4 9}         {0 4 9 4 9}
{1 6.5 6.964 6.…    …964 6.5 6.964}
{2 6.732 6.736 …    …36 6.732 6.736}
      6.732097069         6.732097069
      6.736096793         6.736096793
             Done                Done
```

Bonne réussite!

Pack et mise en page par Laurae, rédigé par critor sur le forum, administrateurs de TI-Planet.org

Correction 10 : Correction exercice algo Maths BAC S 2012 France Métropole

de **critor** » 21 Juin 2012 13:10

Et voici donc le sujet d'algorithmique du BAC S 2012 tombé ce matin en France métropolitaine:

Partie B

Soit (u_n) la suite définie pour tout entier strictement positif par $u_n = 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n} - \ln n$.

1. On considère l'algorithme suivant :

Variables :	i et n sont des entiers naturels. u est un réel.
Entrée :	Demander à l'utilisateur la valeur de n.
Initialisation :	Affecter à u la valeur 0.
Traitement :	Pour i variant de 1 à n. | Affecter à u la valeur $u + \frac{1}{i}$.
Sortie :	Afficher u.

Donner la valeur exacte affichée par cet algorithme lorsque l'utilisateur entre la valeur $n = 3$.

2. Recopier et compléter l'algorithme précédent afin qu'il affiche la valeur de u_n lorsque l'utilisateur entre la valeur de n.

3. Voici les résultats fournis par l'algorithme modifié, arrondis à 10^{-3}.

n	4	5	6	7	8	9	10	100	1000	1500	2000
u_n	0,697	0,674	0,658	0,647	0,638	0,632	0,626	0,582	0,578	0, 578	0,577

À l'aide de ce tableau, formuler des conjectures sur le sens de variation de la suite (u_n) et son éventuelle convergence.

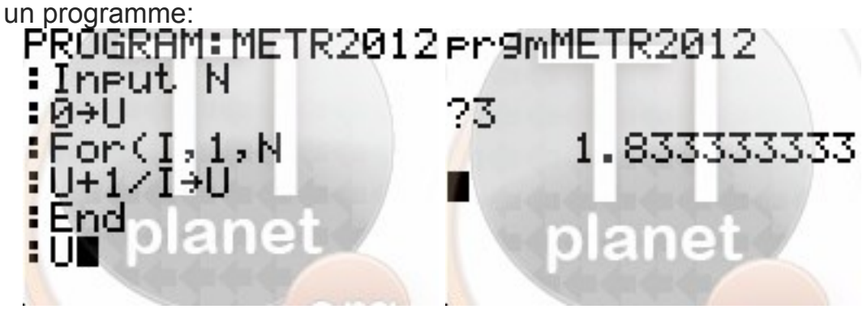
Comme prédit, la question d'algo est ici couplée avec des suites.

La question 1 demande donc la valeur exacte retournée par l'algorithme quand n=3.

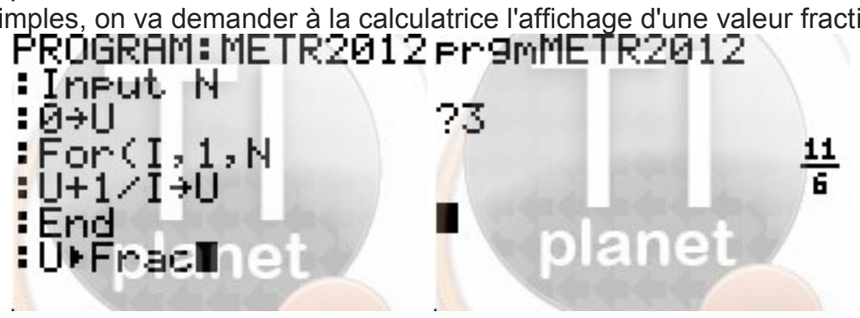
Il suffit de dérouler l'algorithme et d'en réaliser une trace - c'est-à-dire de préciser l'état des variables à chaque étape importante de l'algorithme:

n	i	u	Commentaire
3	non initialisé	0	Fin d'initialisation - juste avant le démarrage de la boucle pour
3	1	1	1ère itération de la boucle pour avec i=1
3	2	1+1/2=3/2	2ème itération de la boucle pour avec i=2
3	3	3/2+1/3=11/6	3ème et dernière itération de la boucle pour avec i=3

La calculatrice peut ici grandement nous aider et nous donner directement le résultat.
Prenons une TI-73 à TI-84 et traduisons l'algorithme en un programme:

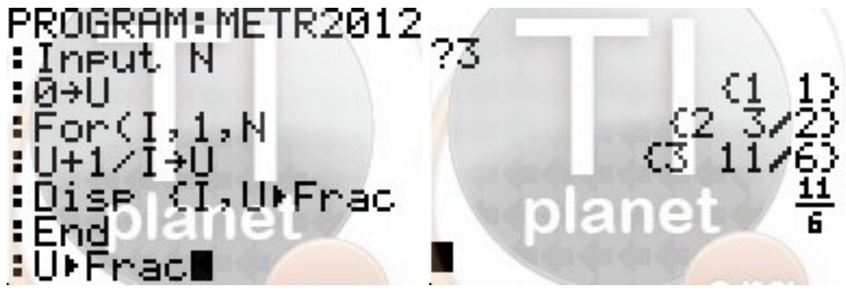


Mais hélas, la calculatrice répond une valeur décimale approchée et non une valeur exacte...
Pas de problème: comme les calculs sont relativement simples, on va demander à la calculatrice l'affichage d'une valeur fractionnaire exacte:



La calculatrice peut même nous donner la justification avec la trace réalisée ci-dessus:

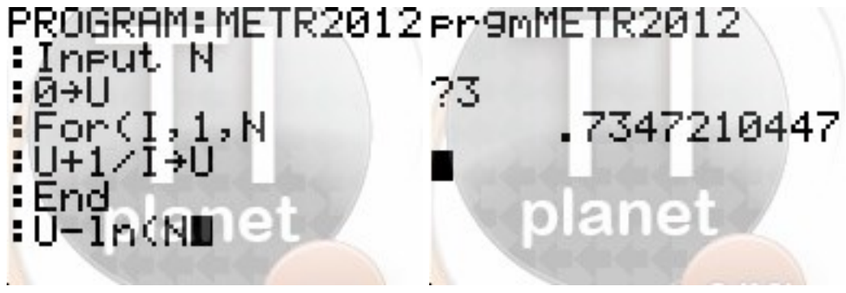
Pack et mise en page par Laurae, rédigé par critor sur le forum, administrateurs de TI-Planet.org



En question 2, on aimerait retourner la valeur de la suite u_n.
Par rapport à la définition générale donnée pour la suite u_n, il manque juste à soustraire ln(n) à la fin.

Modifions donc la dernière ligne de l'algorithme avec:
Afficher u-ln(n).

La calculatrice nous permet de vérifier si l'on a juste.
Modifions le programme:

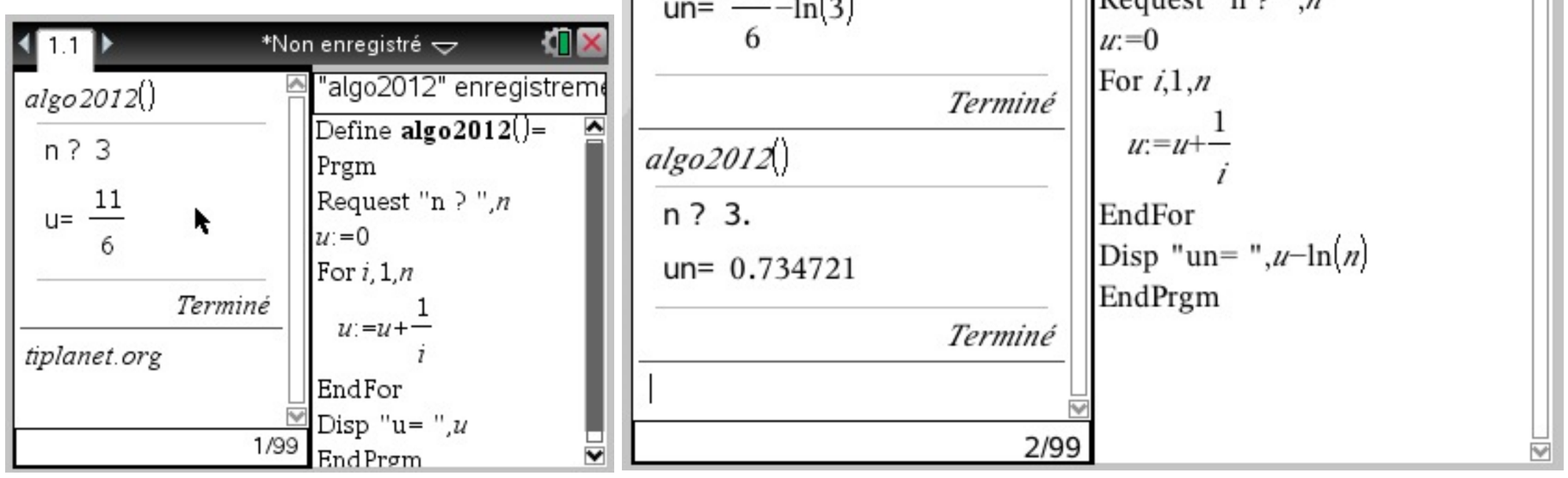


Définissons la suite, et vérifions avec le tableau de valeurs si le résultat du programme est le bon pour n=3:

Le résultat est validé!

Avec le tableau de valeurs de la question 3, on peut conjecturer une décroissance, et une convergence vers une valeur comprise entre 0 et 0,577.

Bien évidemment, une TI-Nspire aurait pu tout aussi bien nous aider:



A bientôt!

Pack et mise en page par Laurae, rédigé par critor sur le forum, administrateurs de TI-Planet.org

Correction 11 : L'algorithme du BAC S d'Antilles-Guyane septembre 2012

de **critor** » 13 Jan 2013 18:02

Tes annales de BAC 2013 papier sont éditées au mois d'août 2012.
Elles ne contiennent donc pas les sujets qui sont tombés aux sessions de remplacement en septembre 2012 pour ceux qui ont manqué des épreuves pour un motif légitime.

Intéressons-nous ce soir au sujet de maths du BAC S qui est tombé en Antilles-Guyane le 13 septembre 2012.

La totalité des sujets de maths du BAC S 2012 tombés en France métropolitaine et dans les centres d'examens à l'étranger à une unité près jusqu'en juin dernier contenaient une ou deux questions d'algorithmique.

Ce nouveau sujet ne fait pas exception. L'algorithmique est à la mode en ce moment au BAC S, tout comme les QCM ou ROC l'ont été en leur temps il y a quelques années.

Alors certes, ce sujet n'est pas conforme au nouveau programme du BAC 2013 et beaucoup de choses sont hors programme. Mais en maths, nous avons eu la chance d'avoir eu le droit à un programme transitoire en 2011-2012, et les écarts avec le nouveau programme sont donc moindres.
Les sujets 2011-2012 restent donc utilisables pour préparer le BAC 2013, à condition bien sûr d'être capable de distinguer ce qui est encore au programme de ce qui ne l'est plus. Et notamment, ce que je vais présenter ci-dessous est toujours au programme.

L'exercice 4 et dernier exercice, mélange ici fonctions et suites et se termine par une question d'algorithmique que voici:

3. On donne l'algorithme suivant :

```
X est une variable réelle ; Y est une variable entière
Affecter 5 à X et 0 à Y
Tant que X > 2,72
    Faire
        Affecter (X / ln X) à X
        Affecter Y + 1 à Y
Fin de Tant que
Afficher Y
```

À l'aide du tableau suivant, obtenu avec un tableur, déterminer la valeur affichée par l'algorithme.

n	0	1	2	3	4	5
u_n	5	3,106 674 672 8	2,740 652 532 3	2,718 372 634 6	2,718 281 830 01	2,718 281 828 5

L'algorithme est donc ici cadeau, et la compétence évaluée est de savoir interpréter/dérouler un algorithme.

Il faut donc comprendre à la lecture qu'il s'agit ici d'un algorithme calculant les termes d'une suite définie par la relation de récurrence $u_{n+1}=u_n/\ln(u_n)$ et par la donnée de son premier terme $u_0=5$.

Dans le corps de l'algorithme, Y joue le rôle de n, et X joue le rôle de u_n.

Il s'agit comme par hasard de la suite étudiée dans le début de l'exercice, et il a été démontré qu'elle était décroissante et convergeait vers e (2,718...).

L'algorithme cherche ici à déterminer à partir de quand rang les termes de la suite seront strictement plus petits que 2,72, c'est-à-dire ici situés dans l'intervalle [e;2,72].

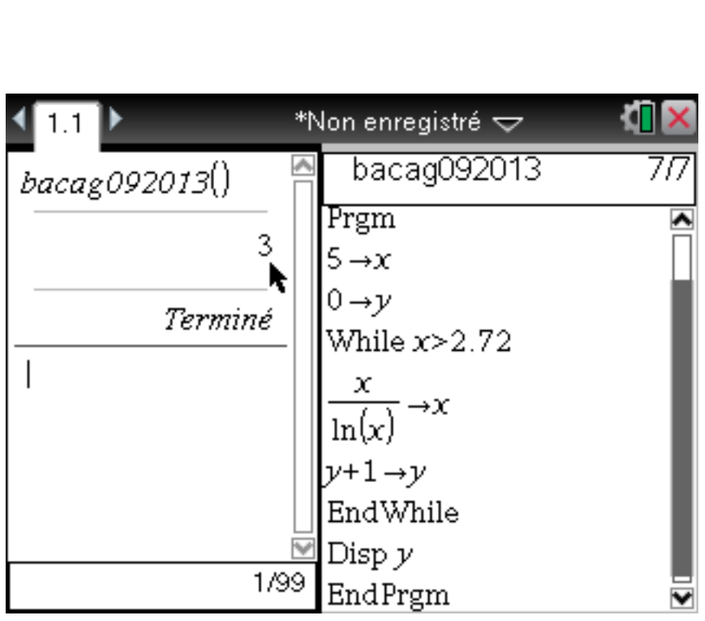
Il peut servir à donner une idée de la vitesse de convergence.

Mais c'était ici une question extrêmement sympa, puisqu'il n'y avait même pas besoin de comprendre tout ça pour y répondre: on nous demande juste ce que renvoie l'algorithme, alors il suffisait 'bêtement' de le traduire en programme pour sa calculatrice comme tu es censé le faire depuis la Seconde et de recopier.

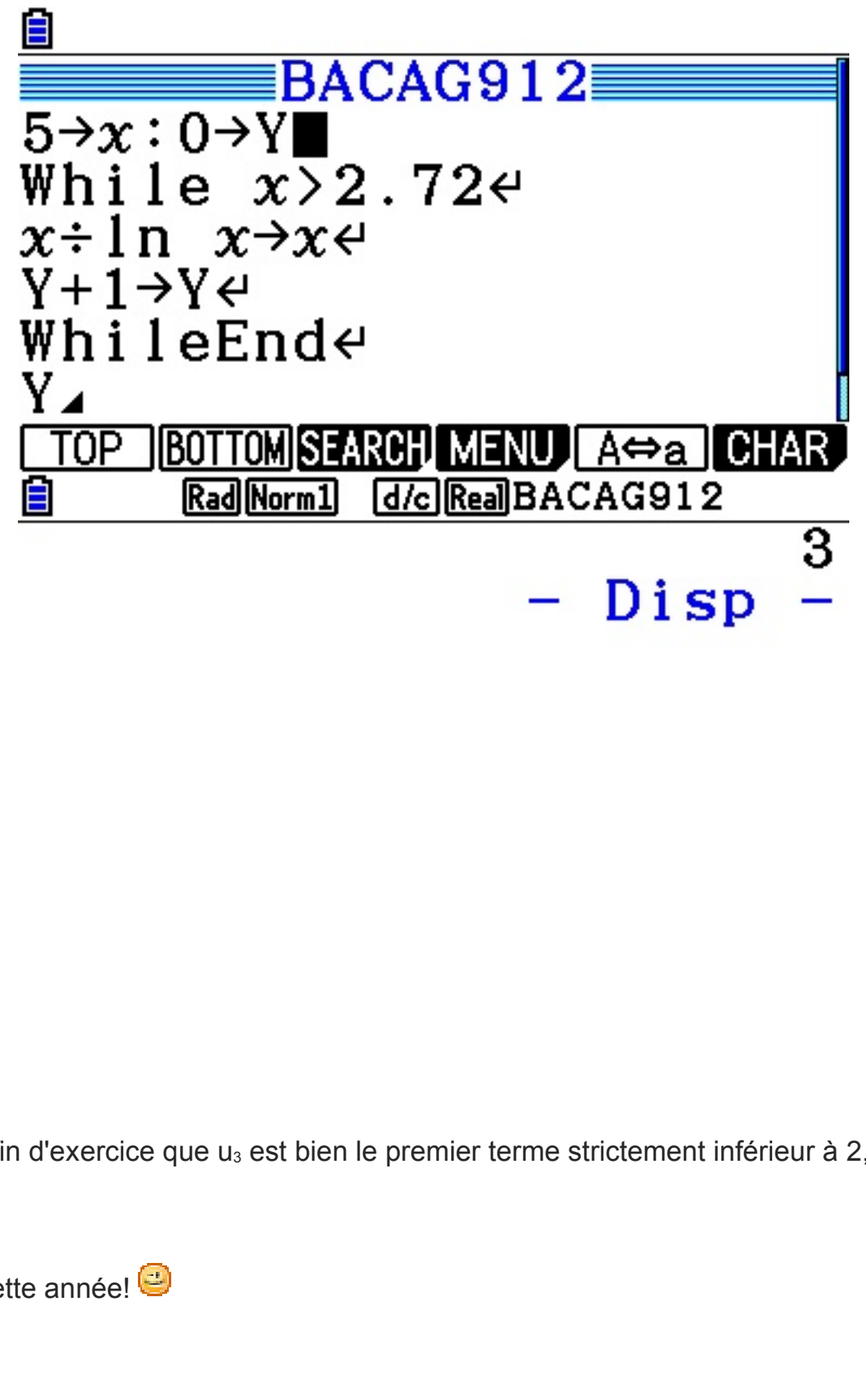
Voici le programme pour TI-76 et TI-82 à TI-84:

```
PROGRAM:BACAG913          prgmBACAG913
:5→X                                 3
:0→Y                              Done
:While X>2.72
:X/ln(X)→X
:Y+1→Y
:End
:Disp Y
```

Ou encore le programme pour TI-Nspire:



Et même, pour ne laisser personne de côté, le programme pour les Casio Graph et Casio Prizm fx-CG:



Bref, dans tous les cas, la réponse est donc 3.

Et effectivement, on vérifie sur le tableau de valeurs en fin d'exercice que u_3 est bien le premier terme strictement inférieur à 2,72.

En te souhaitant une question 'cadeau' du même type cette année!

Correction 12 : Correction 1er algorithme BAC ES-L 2013

de **critor** » 16 Avr 2013 10:49

Dans la news précédente, nous te faisions remarquer que le tout premier sujet de maths du BAC ES-L 2013 pour les candidats du lycée français d'Inde à Pondichéry était tombé avec un algorithme.

Le voici:

3. On donne l'algorithme suivant :

Entrée	Saisir un nombre S supérieur à 3000
Traitement	Affecter à n la valeur 0. {*Initialisation*} Affecter à U la valeur 3000 {*Initialisation*} Tant que $U \leq S$ n prend la valeur $n + 1$ U prend la valeur $U \times 1{,}025$ Fin tant que
Sortie	Afficher le nombre $2000 + n$

a) Pour la valeur S = 3300 saisie, recopier et compléter autant que nécessaire le tableau suivant. Les résultats seront arrondis à l'unité.

Valeur de n	0	1	.....	
Valeur de U	3000		.....	
Condition $U \leq S$	vrai		.....	

b) En déduire l'affichage obtenu quand la valeur de S saisie est 3300.

c) Dans le contexte de cet exercice, expliquer comment interpréter le nombre obtenu en sortie de cet algorithme quand on saisit un nombre S supérieur à 3000.

Question 3)c)
Cet algorithme implémente une suite U définie par:
- U_0=3000
- U_{n+1}=U_n*1,025 (augmentation de 2,5%)

Il s'agit en fait de la suite C définie dans le contexte de l'exercice.

La suite étant ici géométrique de premier terme strictement positif et de raison 1,025 strictement supérieure à 1, c'est une suite strictement croissante.

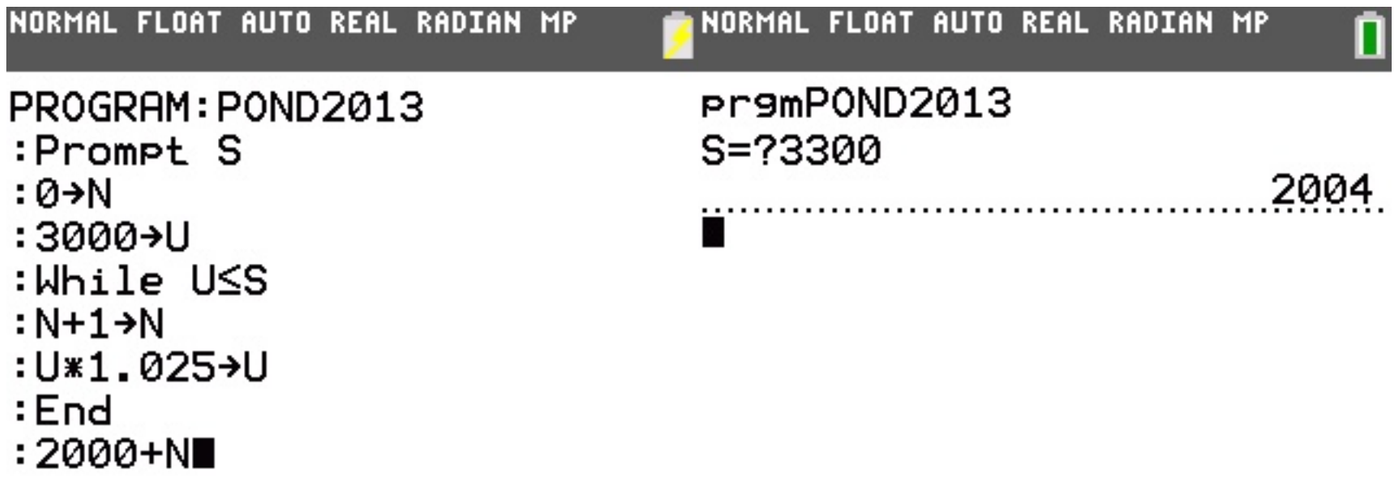
La boucle 'tant que' incrémente l'indice n en partant de 0 et s'arrête lorsque l'on réalise le contraire de U≤S, c'est-à-dire U>S.
En sortie de la boucle, l'indice n correspond donc au premier terme de la suite ayant dépassé la valeur S saisie.

Le but de l'algorithme est donc de retourner la 1ère année où la somme placée aura strictement dépassé la valeur S saisie.

Question 3)b)
Si ce qui précède a été anticipé, il est facile d'obtenir la valeur retournée en demandant un simple tableau de valeurs de la suite C de l'exercice, et en rajoutant 2000 à la valeur d'indice n trouvée.

Sinon, il suffit de traduire l'algorithme en un programme sur sa calculatrice et de le lancer.

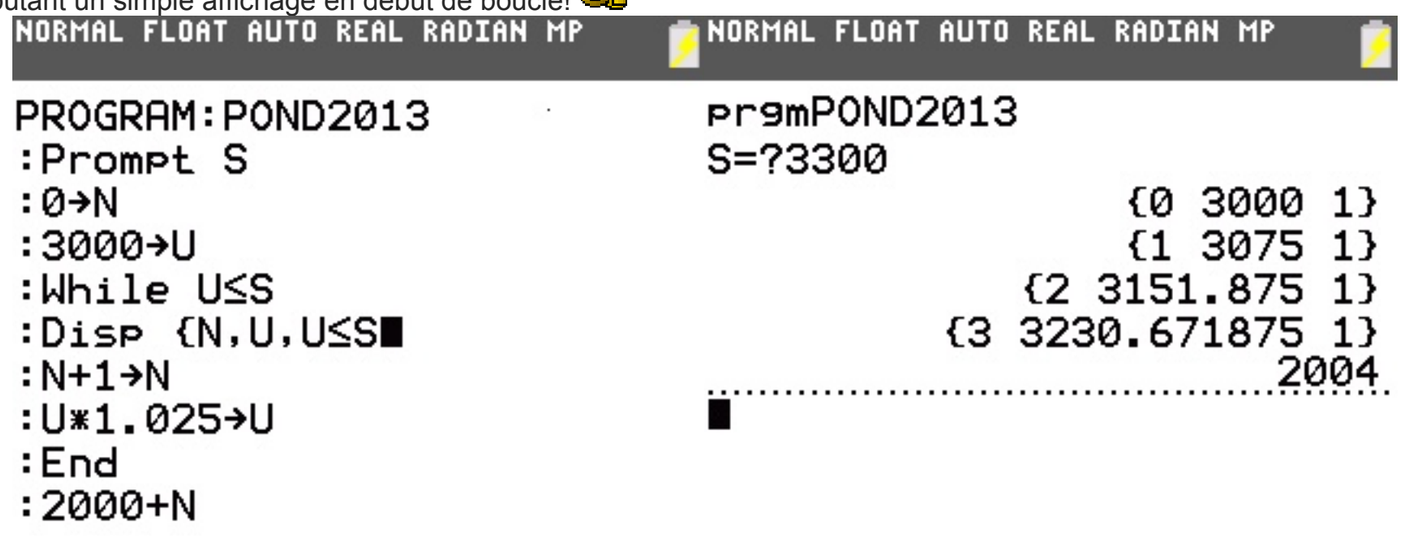
Le voici par exemple pour toutes les calculatrices graphiques TI-73 à TI-84:



La réponse est donc 2004.

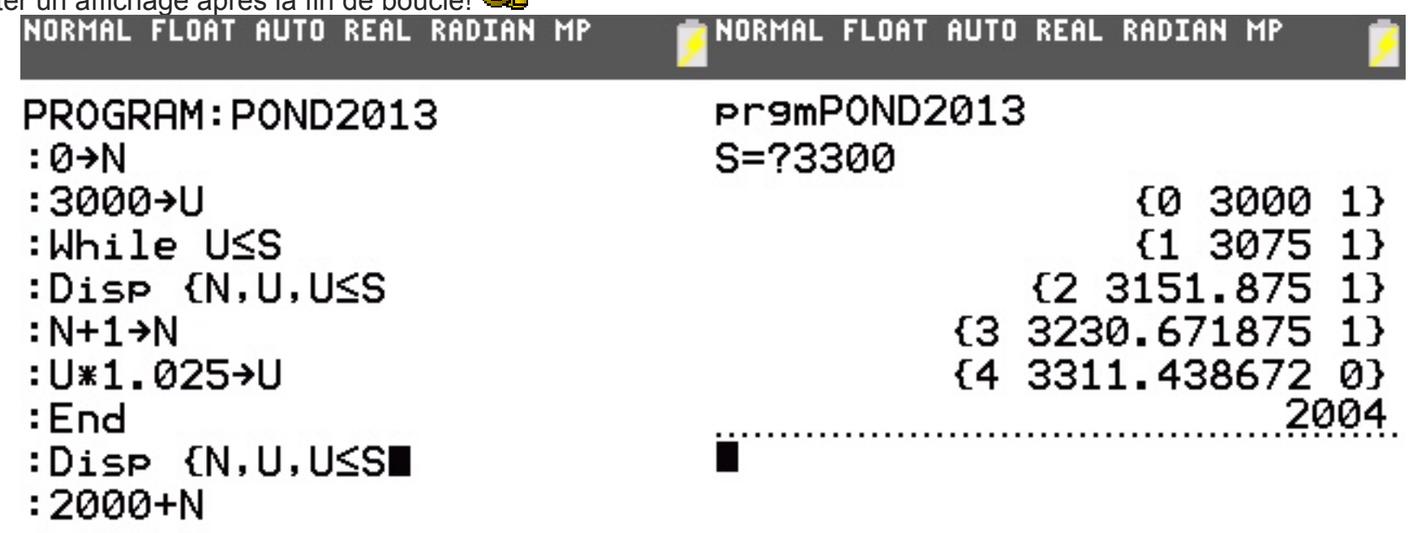
Question 3)a)
On nous demande ici de réaliser une sorte de trace de l'algorithme, avec les états des variables N, U et du test U≤S à chaque étape de l'algorithme.

Il est ultra facile de l'obtenir en rajoutant un simple affichage en début de boucle!



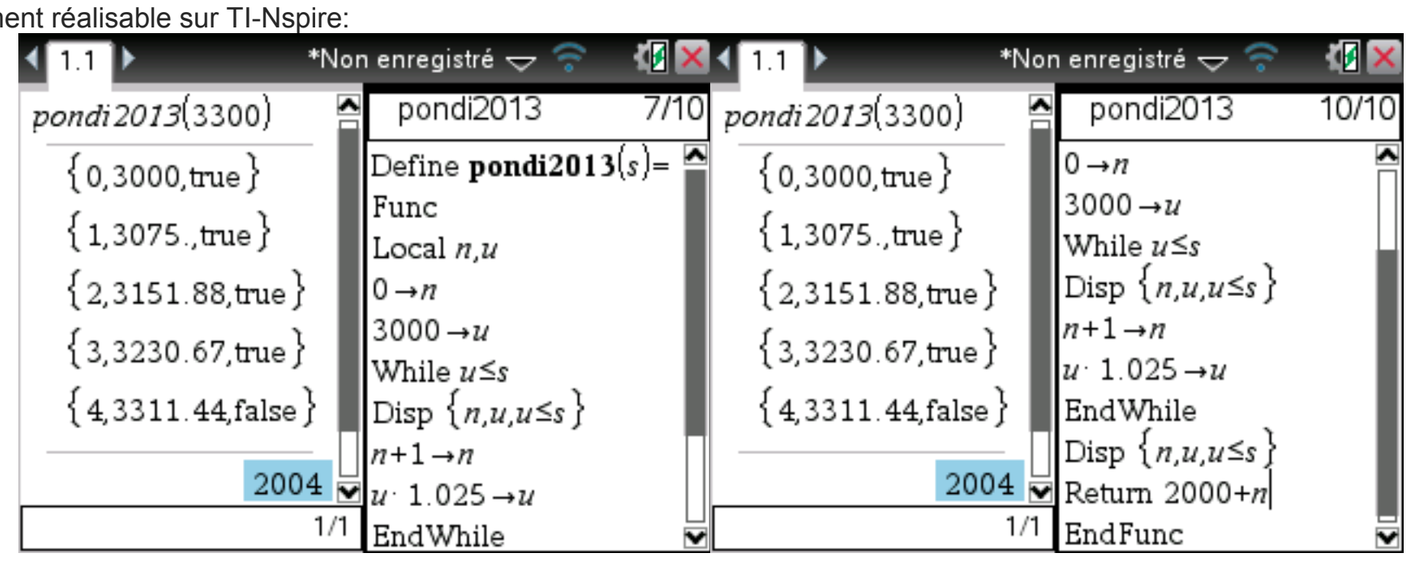
Notons que la calculatrice affiche ici pour le test 1 pour vrai et 0 pour faux.

Mais petit problème, nous n'obtenons pas ici l'état final avec le test faux, puisque dans ce cas on ne passe pas dans la boucle.
Pas de problème, il suffit de rajouter un affichage après la fin de boucle!

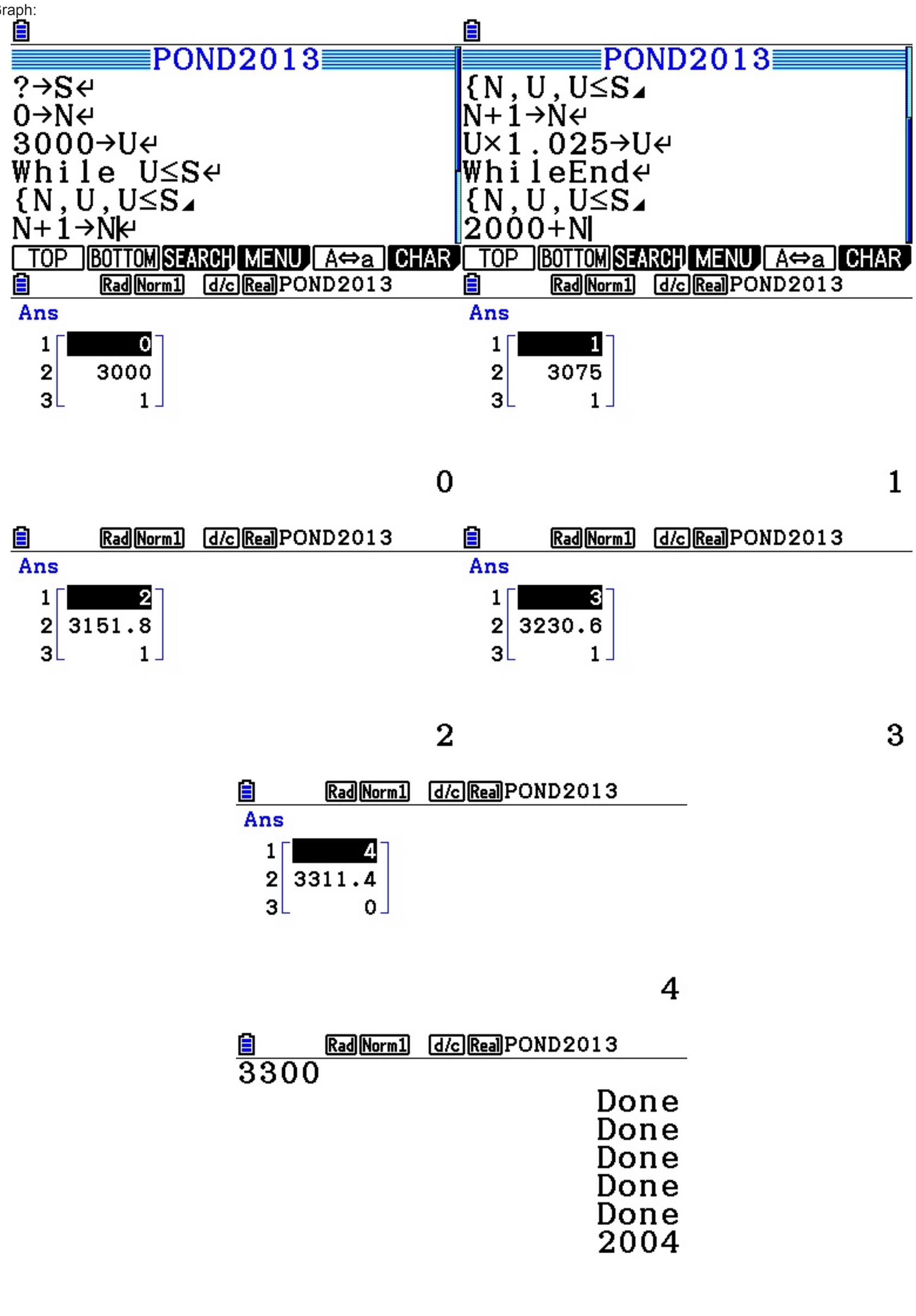


Et voilà, le tableau prêt à être recopié!

La même chose est bien évidemment réalisable sur TI-Nspire:



Ou encore sur Casio Graph:



A bientôt!

Pack et mise en page par Laurae, rédigé par critor sur le forum, administrateurs de TI-Planet.org

Correction 13 : Correction 1er algorithme BAC S 2013 (Pondichéry, Inde)

de **critor** » 16 Avr 2013 22:47

Ce matin vous découvriez donc le 1er sujet de maths du nouveau BAC S 2013, et ça n'a pas raté puisqu'il contient un algorithme.

Variables	K et J sont des entiers naturels, P est un nombre réel
Initialisation	P prend la valeur 0 J prend la valeur 1
Entrée	Saisir la valeur de K
Traitement	Tant que $P < 0{,}05 - 10^{-K}$ P prend la valeur $0{,}2 \times P + 0{,}04$ J prend la valeur J+1 Fin tant que
Sortie	Afficher J

A quoi correspond l'affichage final J ?
Pourquoi est-on sûr que cet algorithme s'arrête ?

Replaçons cette question dans son contexte:
- elle fait partie d'un exercice mélangeant probabilités et suites
- on considère la suite p définie par récurrence par $p_1=0$ et $p_{n+1}=0{,}2p_n+0{,}04$
- on a démontré que cette suite avait pour limite 0,05
- l'énoncé nous dit que cette suite est croissante

De façon fort habituelle, l'algorithme implémente cette suite, les valeurs des termes étant stockés dans la variable P.
On a bien en effet:
- en initialisation: P prend la valeur 0
- P prend la valeur 0,2P+0,04

C'est la variable J, initialisée à 1 et incrémentée de 1, qui joue ici le rôle de l'indice.

Pour comprendre ce que réalise un algorithme constitué d'une boucle tant que, il convient de regarder quand est-ce que cette boucle s'arrête.

Elle s'arrête lorsque l'on obtient le contraire de $P<0{,}05-10^{-K}$, c'est-à-dire $P\geq 0{,}05-10^{-K}$.

Or, on sait de plus que la suite est croissante et a pour limite 0,05. On a donc dans tous les cas $P<0{,}05$.

Pour K=2, la suite s'arrête donc au premier terme P vérifiant $0{,}04\leq P<0{,}05$.
Pour K=3, la suite s'arrête donc au premier terme P vérifiant $0{,}049\leq P<0{,}05$.
Pour K=4, la suite s'arrête donc au premier terme P vérifiant $0{,}0499\leq P<0{,}05$.
Etc...

L'affichage final est alors le rang J associé à la valeur de ce terme.

Cet algorithme permet donc d'étudier la convergence de la suite p vers 0,05, et notamment la 'vitesse' de convergence.
Il renvoie l'indice du premier terme étant aussi près que l'on veut de 0,05 (à 10^{-K} près).

Par définition de la convergence de la suite p vers 0,05 on sait que:
pour tout réel $a>0$, il existe un rang n_0 tel que pour tout $n\geq n_0$, $0{,}05-a<p_0<0{,}05+a$

10^{-K} qui est bien un réel positif joue le rôle du réel a (écart à la limite 0,05).

Par définition, on est sûr que l'algorithme se termine: nous rencontrerons forcément pour un certain rang un terme supérieur à $0{,}05-10^{-K}$ qui interrompt alors la boucle tant que.

Correction 14 : Correction algorithme BAC L 2012 Nouvelle Calédonie

de **critor** » 22 Avr 2013 12:18

Bonjour à tous!

Aujourd'hui nous allons nous intéresser à l'algorithme qui est tombé en novembre 2012 au BAC L de Nouvelle Calédonie, et ce n'est pas pour ça qu'il en est moins intéressant, au contraire!

EXERCICE 2 **(4 points)**

Soient N et B deux nombres entiers naturels. On appelle Q le quotient entier de N par B.
Par exemple si $B=3$ et $N=17$ alors le quotient entier de N par B est $Q=5$.
On considère l'algorithme :

Entrée	Deux entiers naturels N et B
Initialisation	$Q=1$
Traitement	Tant que $Q \neq 0$. Affecter à Q le quotient entier de N par B. Affecter à R la valeur $(N-Q\times B)$. Afficher R. Affecter à N la valeur de Q.

1) Quelles sont les valeurs successives de R affichées par cet algorithme pour $B=2$ et $N=12$?

2) a) Quelles sont les valeurs successives de R affichées par cet algorithme pour $B=3$ et $N=346$?
b) Quelle est l'écriture de 346 en base 3.

3) Que permet de déterminer cet algorithme ?

Bon avant d'aller plus loin, il se trouve que cet algorithme est manifestement faux puisque la boucle 'Tant que' n'est pas fermée.
Si on le traduit tel quel sur calculatrice, les comportements seront très variés. Selon les modèles nous pourrons obtenir une erreur ou un comportement différent de celui attendu.

Je vous propose donc la correction suivante:

EXERCICE 2 **(4 points)**

Soient N et B deux nombres entiers naturels. On appelle Q le quotient entier de N par B.
Par exemple si $B=3$ et $N=17$ alors le quotient entier de N par B est $Q=5$.
On considère l'algorithme :

Entrée	Deux entiers naturels N et B
Initialisation	$Q=1$
Traitement	Tant que $Q \neq 0$. Affecter à Q le quotient entier de N par B. Affecter à R la valeur $(N-Q\times B)$. Afficher R. Affecter à N la valeur de Q. Fin Tant que

1) Quelles sont les valeurs successives de R affichées par cet algorithme pour $B=2$ et $N=12$?

2) a) Quelles sont les valeurs successives de R affichées par cet algorithme pour $B=3$ et $N=346$?
b) Quelle est l'écriture de 346 en base 3.

3) Que permet de déterminer cet algorithme ?

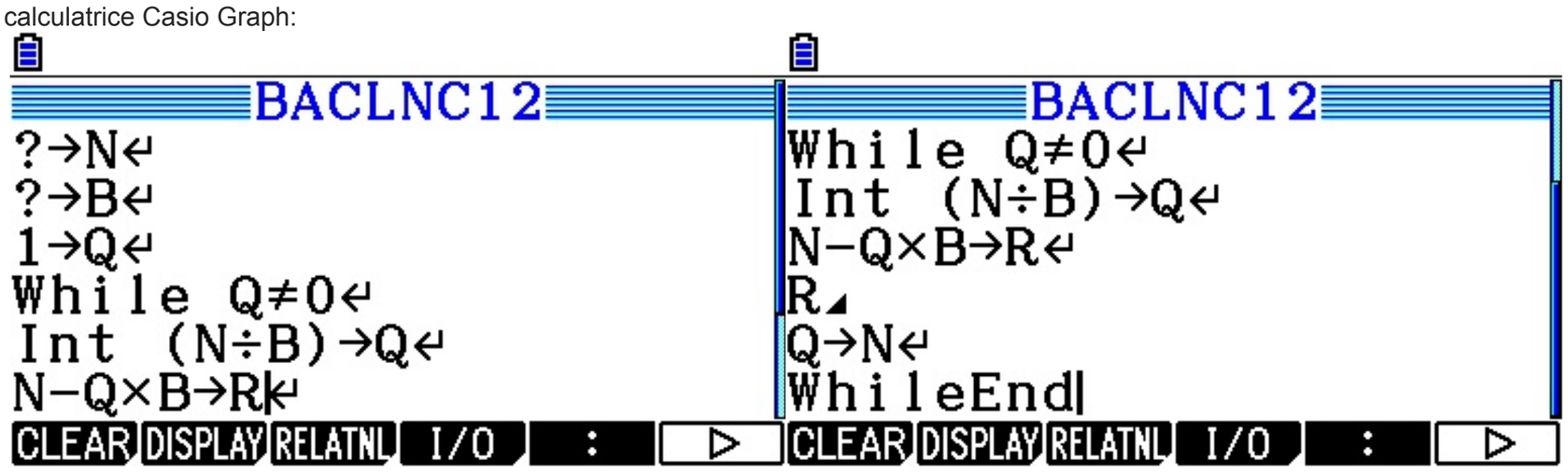
Question 1:

Pack et mise en page par Laurae, rédigé par critor sur le forum, administrateurs de TI-Planet.org

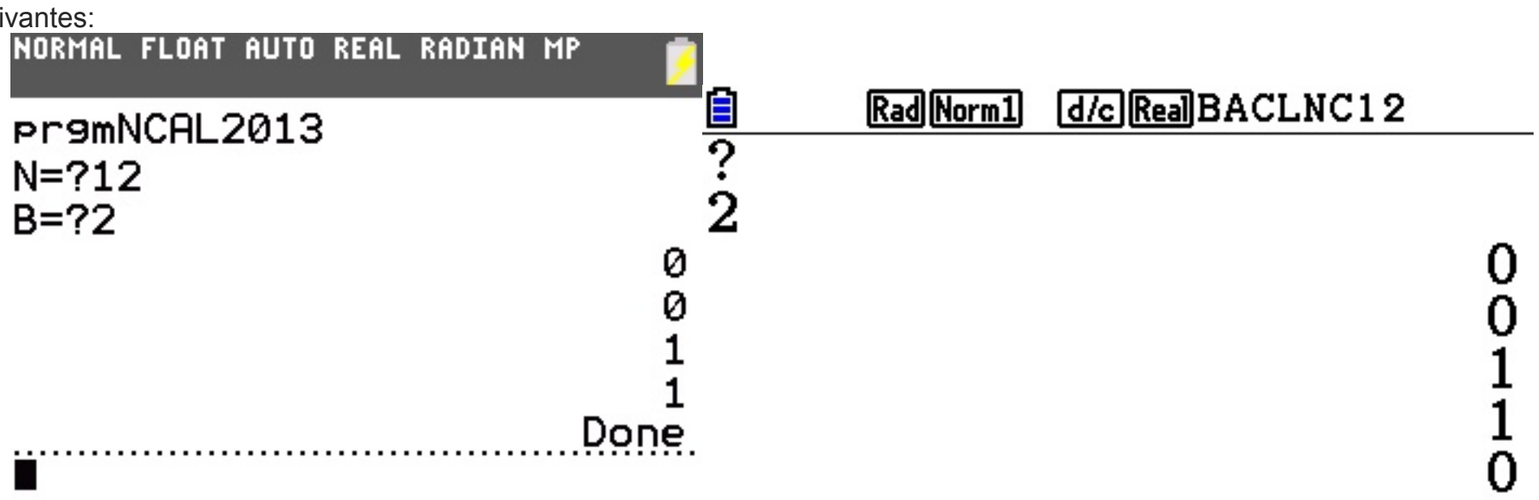
On nous demande donc dès le départ de donner les sorties de l'algorithme lorsque B=2 et N=12.
Le plus simple est donc de traduire cet algorithme en un programme pour nos calculatrices graphiques.

Le voici pour calculatrice graphique TI-73 à TI-84:

Ou le voici encore pour calculatrice Casio Graph:



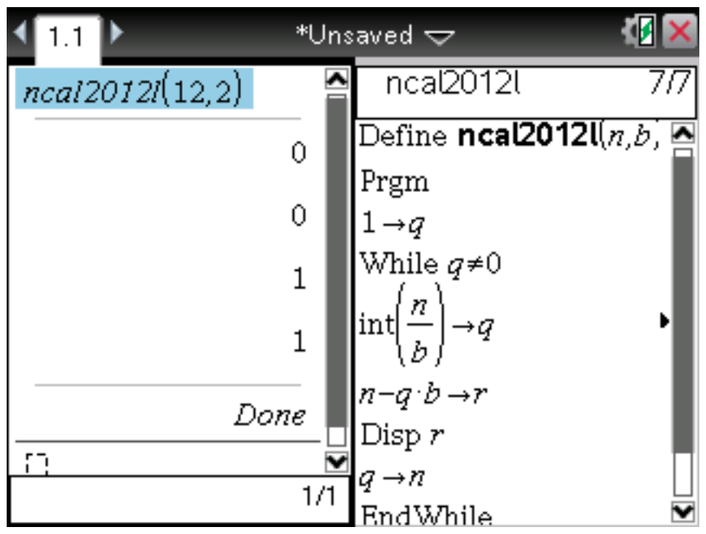
Nous obtenons les sorties suivantes:



Notre TI-84 Plus C Silver Edition affiche ici 0011 et notre Casio Prizm fx-CG20 00110.

Sur Casio Prizm, le dernier 0 n'est pas un affichage correspond en fait au résultat de la dernière instruction, l'affectation de la variable N avec la valeur 0 de Q. Il remplace en fait le "Done" de la TI-84.
Bref, il ne compte pas et la sortie est donc bien 0011 dans les deux cas.

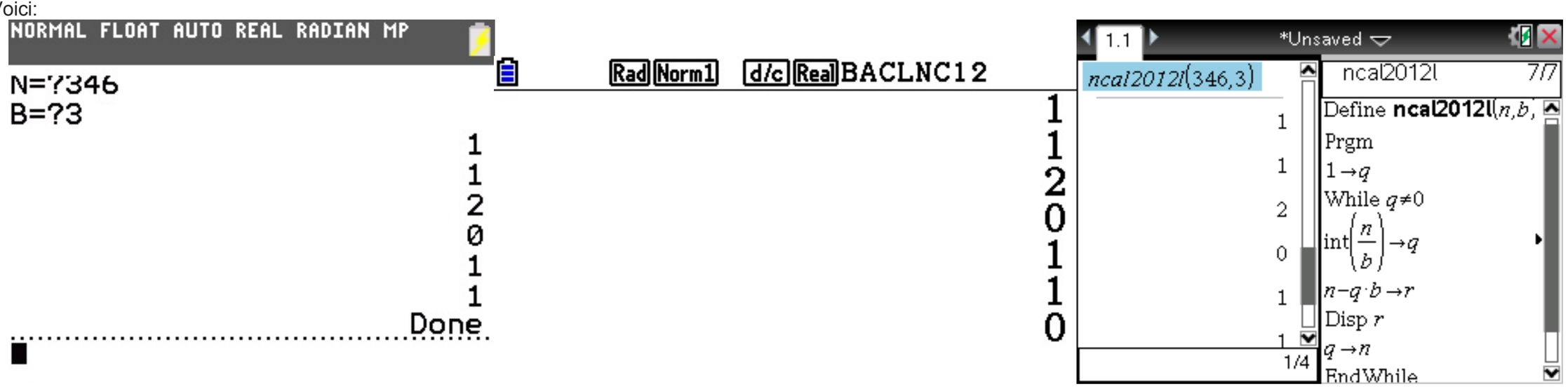
La même chose est réalisable sur TI-Nspire:



Question 2a:
On nous demande maintenant de recommencer avec B=3 et n=346.

Pack et mise en page par Laurae, rédigé par critor sur le forum, administrateurs de TI-Planet.org

Voici:



La réponse est donc 112011.

Question 2b:
Comme par hasard, on nous demande maintenant l'écriture de 346 en base 3, mêmes données donc que la question précédente.
Il se trouve que c'est 110211, et l'on peut remarquer que c'est donc l'envers de notre réponse à la question précédente.

Question 3:
L'exemple de la question précédente nous permet de répondre ici: le rôle de l'algorithme est de donner, de droite à gauche, l'écriture du nombre N en base B.

A bientôt!

Pack et mise en page par Laurae, rédigé par critor sur le forum, administrateurs de TI-Planet.org

Correction 15 : Correction algorithme BAC L France septembre 2012

de **critor** » 22 Avr 2013 15:21

Et nous revoici pour décortiquer devant vous un nouvel algorithme du BAC, cette fois-ci celui tombé au BAC L 2012 en France à la session de remplacement de septembre 2012.

Exercice 1 *(5 points)*

1. On considère l'algorithme suivant :

Entrée :	A est un nombre entier naturel.
Traitement :	Affecter à N la valeur A^2-3A+6 Tant que $N \geq 0$ Affecter à N la valeur $N-4$
Sortie :	Afficher la valeur de N.

a) Que donne l'affichage en sortie pour chacune des entrées suivantes : $A=5$, $A=8$ et $A=9$?

b) Modifier l'algorithme pour que, avec l'entier A pour entrée, la valeur affichée en sortie donne le reste de la division euclidienne de l'entier A^2-3A+6 par 4.

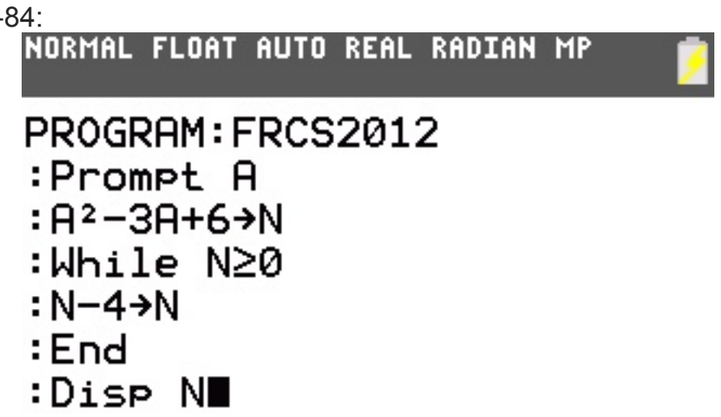
Là encore, la fin de boucle tant que est omise, mais la différence est que l'on a l'indentation qui ne permet donc aucune ambiguïté.

Question 1a:

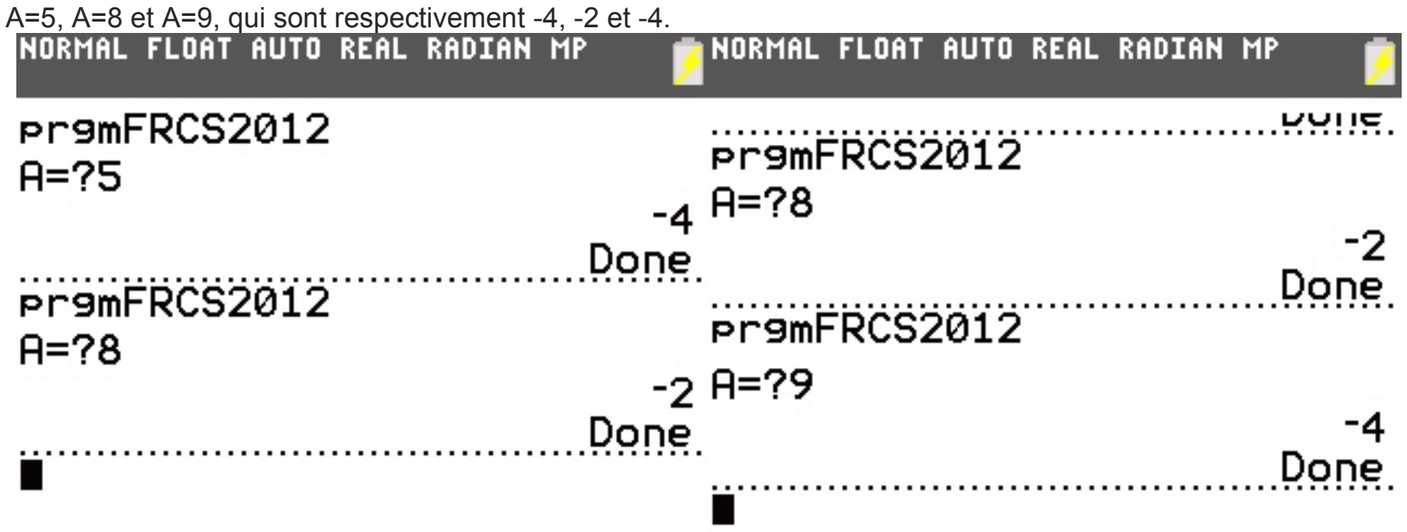
On nous demande donc l'affichage de l'algorithme pour différentes valeurs de l'entrée A.

Il nous suffit de le traduire en un programme de quelques lignes pour nos calculatrices graphiques et nous aurons la réponse sans effort!

Le voici par exemple pour calculatrices graphiques TI-73 à TI-84:



Et suivent donc les réponses pour A=5, A=8 et A=9, qui sont respectivement -4, -2 et -4.



Question 1b:

Il nous faudrait donc maintenant modifier l'algorithme pour qu'il affiche le reste de la division euclidienne de A²-3A+6 par 4, deux nombres effectivement utilisés par l'algorithme.

Il serait déjà utile de connaître les valeurs du nombre A²-3A+6 afin d'en déduire le reste attendu, et également ce qui ne va pas dans l'algorithme actuel.

Commençons donc par modifier le programme afin d'afficher le nombre A²-3A+6:

Pack et mise en page par Laurae, rédigé par critor sur le forum, administrateurs de TI-Planet.org

```
NORMAL FLOAT AUTO REAL RADIAN MP
PROGRAM:FRCS2012
:Prompt A
:A²-3A+6→N
:Disp N
:While N≥0
:N-4→N
:End
:Disp N
```

Voyons donc maintenant ce que nous obtenons:

```
NORMAL FLOAT AUTO REAL RADIAN MP
H=?5
                16
                -4
              Done
prgmFRCS2012
A=?8
                46
                -2
              Done
```

```
NORMAL FLOAT AUTO REAL RADIAN MP
H=?8
                46
                -2
              Done
prgmFRCS2012
A=?9
                60
                -4
              Done
```

Pour A=5, A²-3A+6=16 et le nombre affiché est -4.
Pour A=8, A²-3A+6=46 et le nombre affiché est -2.
Pour A=9, A²-3A+6=60 et le nombre affiché est -4.

On remarque que le nombre N affiché est toujours strictement négatif, ce qui est normal puisque la boucle TantQue se termine sur la condition N<0.

Or, un reste de division euclidienne par 4 est par définition un nombre entier compris entre 0 et 3.

Il y a ici une itération de trop dans la boucle qui soustrait 4, ce qui rend le résultat négatif.

Deux solutions sont envisageables pour corriger cela.

On peut par exemple tout simplement annuler l'itération supplémentaire en ajoutant 4 au résultat:
Afficher la valeur de N+4

```
NORMAL FLOAT AUTO REAL RADIAN MP
PROGRAM:FRCS2012
:Prompt A
:A²-3A+6→N
:Disp N
:While N≥0
:N-4→N
:End
:Disp N+4
```

On bien, on change le test de la boucle afin qu'elle se termine une itération plus tôt, ce qui est une meilleure solution algorithmique puisqu'on évite des calculs inutiles à la machine:
Tant que N≥4

```
NORMAL FLOAT AUTO REAL RADIAN MP
PROGRAM:FRCS2012
:Prompt A
:A²-3A+6→N
:Disp N
:While N≥4
:N-4→N
:End
:Disp N
```

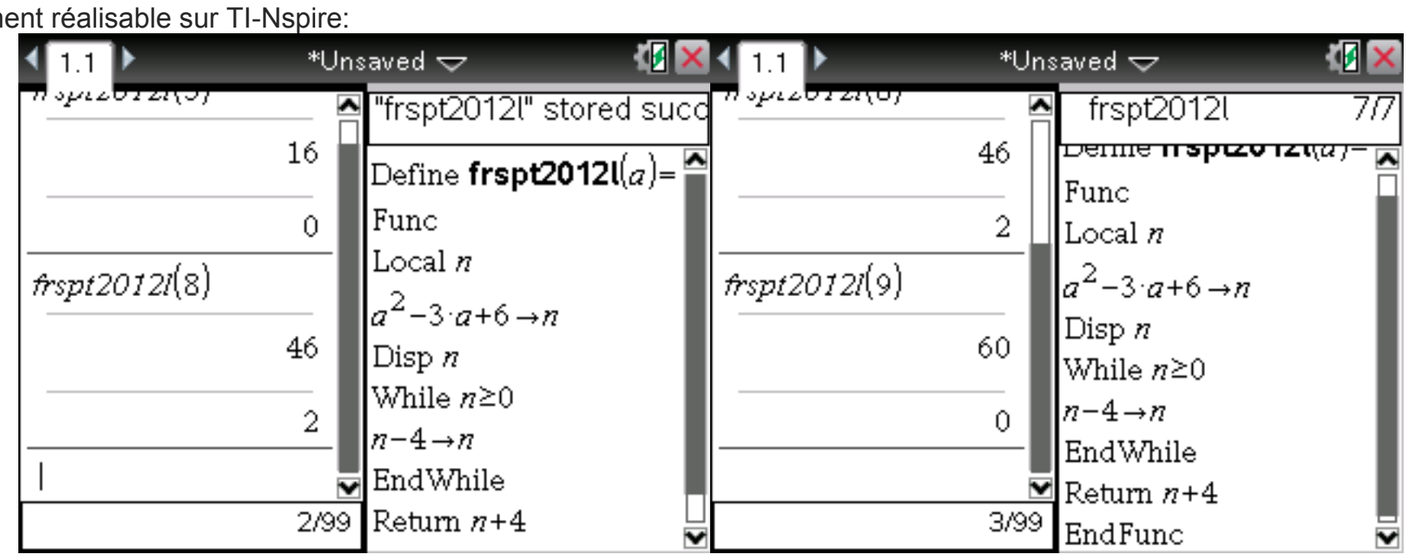
Dans les deux cas, on obtient désormais les restes attendus, respectivement 0, 2 et 0:

Pack et mise en page par Laurae, rédigé par critor sur le forum, administrateurs de TI-Planet.org

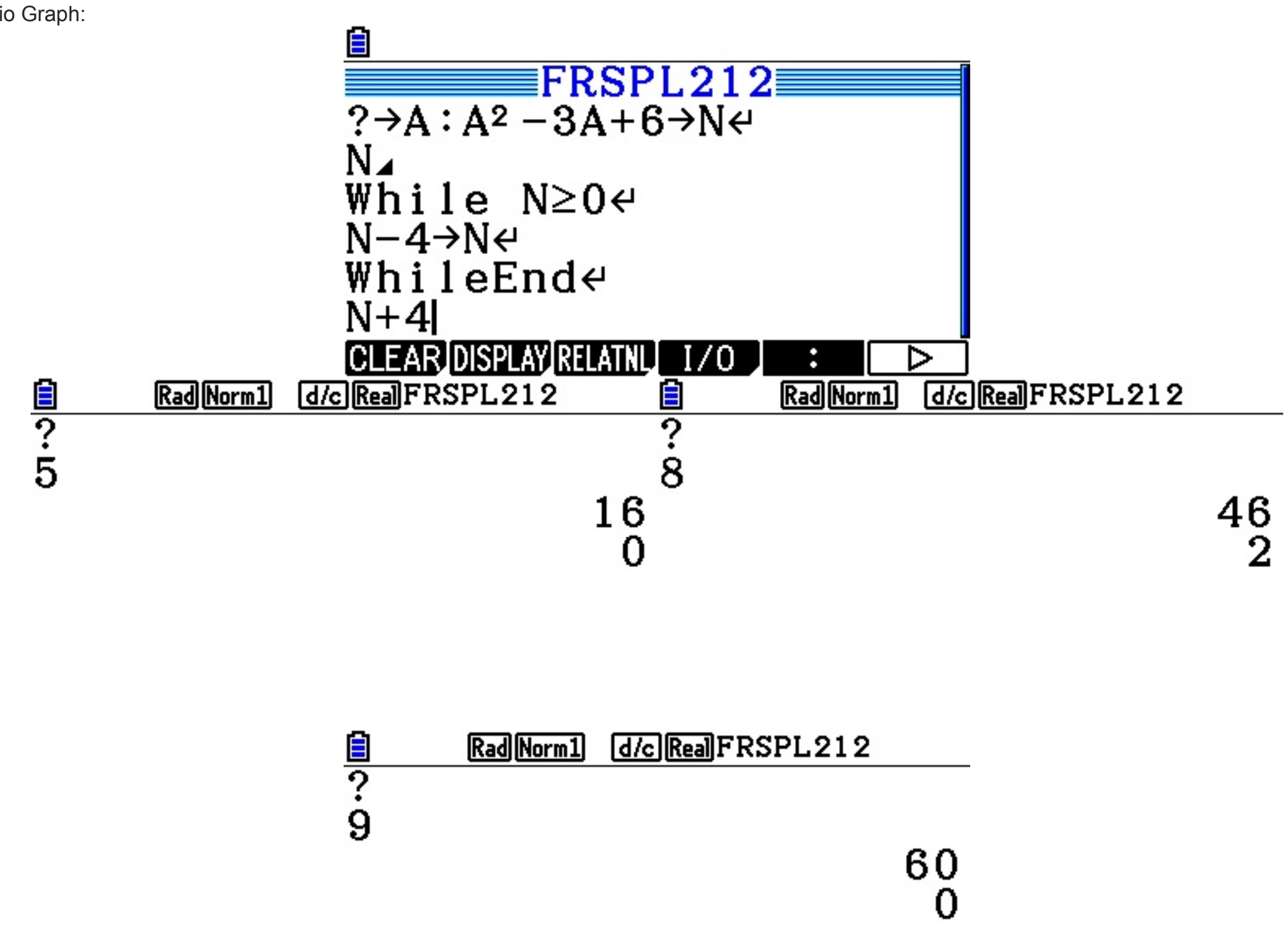
```
NORMAL FLOAT AUTO REAL RADIAN MP
H=?5
                                16
                                 0
                              Done
prgmFRCS2012
A=?8
                                46
                                 2
                              Done
```

```
NORMAL FLOAT AUTO REAL RADIAN MP
H=?8
                                46
                                 2
                              Done
prgmFRCS2012
A=?9
                                60
                                 0
                              Done
```

La même chose est bien évidemment réalisable sur TI-Nspire:



... ou encore sur Casio Graph:



Pack et mise en page par Laurae, rédigé par critor sur le forum, administrateurs de TI-Planet.org

Correction 16 : Correction algorithme BAC S France septembre 2012

de **critor** » 22 Avr 2013 16:55

Nous revoilà encore avec cette fois-ci l'algorithme qui est tombé au BAC S 2012 en France, à la session de remplacement de septembre 2012 :

4. On définit la suite (d_n) par : $d_0 = 1$ et pour tout entier naturel n, $d_{n+1} = \frac{1}{2}d_n^2$.

a. Démontrer par récurrence que pour tout entier naturel n,

$$u_n - \sqrt{7} \leqslant d_n.$$

b. Voici un algorithme :

Variables :	n et p sont des entiers naturels.
	d est un réel.
Entrée :	Demander à l'utilisateur la valeur de p.
Initialisations :	Affecter à d la valeur 1.
	Affecter à n la valeur 0.
Traitement :	Tant que $d > 10^{-p}$.
	Affecter à d la valeur $0,5d^2$.
	Affecter à n la valeur $n+1$.
Sortie :	Afficher n.

En entrant la valeur 9, l'algorithme affiche le nombre 5.
Quelle inégalité peut-on en déduire pour d_5 ?
Justifier que u_5 est une valeur approchée de $\sqrt{7}$ à 10^{-9} près.

C'est donc une situation type en série S, où l'algorithme tombe souvent dans le contexte de suites définies par récurrence.

Comme souvent dans ce cas, l'algorithme réimplémente la suite à l'aide d'une boucle et de deux variables.

La variable n jour le rôle de l'indice, comme le montrent les instructions suivantes d'initialisation et d'incrémentation:
- Affecter à n la valeur 0
- Affecter à n la valeur n+1

On a d'une part pour la suite:
- $d_0=1$
- $d_{n+1}=0,5d_n^2$

Et on retrouve ces deux mêmes informations dans l'algorithme:
- Affecter à d la valeur 1
- Affecter à d la valeur $0,5d^2$

La variable d prend donc les valeurs des termes de la suite d_n.

Première question:
On entre donc la valeur 9 pour p et l'algorithme nous répond 5.

Cela veut donc dire qu'en sortie de l'algorithme on a n=5.

Mais si l'algorithme se termine, c'est que la boucle 'tant que' s'est terminée sur la réalisation du contraire de $d>10^{-p}$, c'est-à-dire $d\leq 10^{-p}$.

Comme p=9 et comme on termine avec n=5, on en déduit l'inégalité $d_5\leq 10^{-9}$.

Deuxième question:
On sait donc que $d_5\leq 10^{-9}$.

Or, on a montré au 4)a) que pour tout entier $n\geq 0$, $u_n-\sqrt{7}\leq d_n$.
On en déduit donc pour n=5, $u_5-\sqrt{7}\leq d_5$.

On obtient ainsi par transitivité $u_5-\sqrt{7}\leq 10^{-9}$, soit $u_5\leq \sqrt{7}+10^{-9}$.

Or, d'après 1)b) on sait que pour tout entier $n\geq 0$, $u_n\geq\sqrt{7}$.
Donc pour n=5, $u_5\geq\sqrt{7}$

On a donc l'encadrement $\sqrt{7}\leq u_5\leq\sqrt{7}+10^{-9}$.

u_5 est donc bien une valeur approchée de $\sqrt{7}$ à 10^{-9} près.

Pack et mise en page par Laurae, rédigé par critor sur le forum, administrateurs de TI-Planet.org

Correction 17 : Correction algorithme BAC S en Amérique du Sud novembre 2012

de **critor** » 26 Avr 2013 18:37

Bonjour!

Aujourd'hui nous allons jeter un coup d'oeil sur un dernier algorithme que nous n'avions pas corrigé jusqu'à présent, celui tombé à l'épreuves de maths du BAC S dans les lycées français d'Amérique du Sud:

3. On donne l'algorithme suivant :

Entrée	Affecter la valeur 3 à la variable n.
Traitement	Tant que $f(n) > 0,1$
	incrémenter la variable n de 1.
	Fin Tant que
Sortie	Afficher la valeur de n.

où f est la fonction étudiée dans la **partie A**.

a. À l'aide de la question 2. a. de la **partie A**, expliquer pourquoi il est certain que cet algorithme donne une valeur en sortie.

b. Quelle est la valeur n_0 de la variable n obtenue à la sortie de l'algorithme ?

c. L'absorption du médicament par l'animal a lieu un matin à 8 h. À quelle question cet algorithme permet-il de répondre ?

Pour une fois, l'algorithme ne tombe pas dans le contexte de suites mais de fonctions.

La fonction f ici mentionnée a pour expression $f(x)=x/2*e^{-x/2}$ et représente la quantité en mg de principe actif d'un médicament passée dans le sans en fonction du temps, le temps zéro correspondant ici à l'injection.

Question B)3)a):
L'algorithme est constitué ici d'une boucle 'tant que'.
Sa condition de poursuite est f(n)>0,1.

Si l'algorithme s'arrête, c'est que l'on a réalisé la condition contraire: $f(n)\leq 0,1$.

n est ici une variable initialisée à 3 et incrémentée de 1 par la boucle.

Cette boucle teste donc les images par la fonction f pour n=3, 4, 5, 6, etc...

Or, on a montré au A)2)a) que la fonction f avait pour limite 0 en $+\infty$.

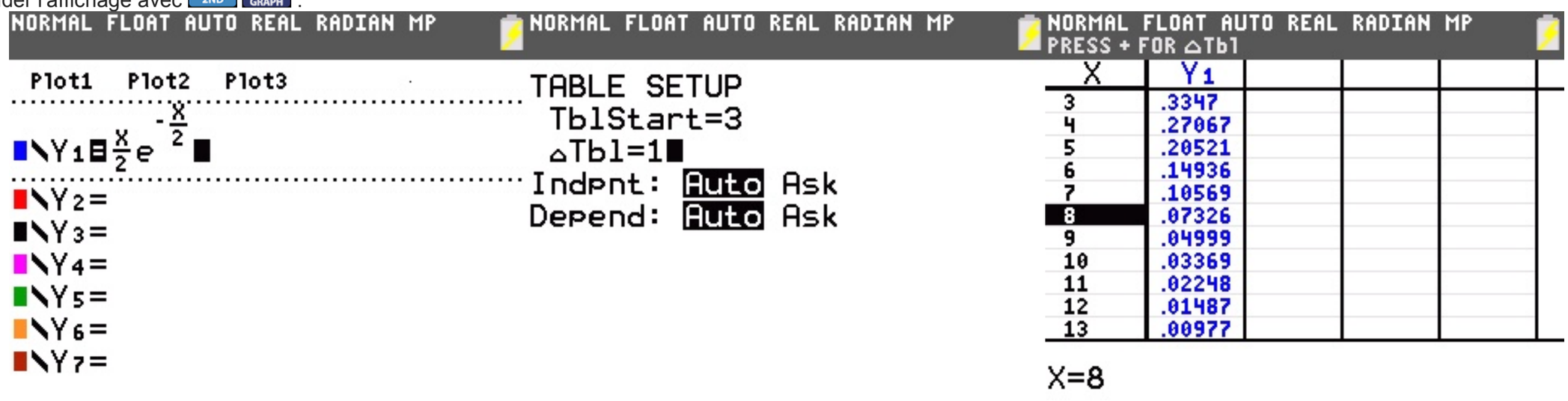
Par définition même de cette limite, pour tout réel a>0, il existe un certain rang n_0 tel que pour tout $n\geq n_0$, $-a\leq f(n)\leq a$.

En prenant a=0,1, il existe un certain rang n_0 tel que pour tout $n\geq n_0$, $-0,1\leq f(n)\leq 0,1$.

Donc forcément, l'algorithme s'arrête.

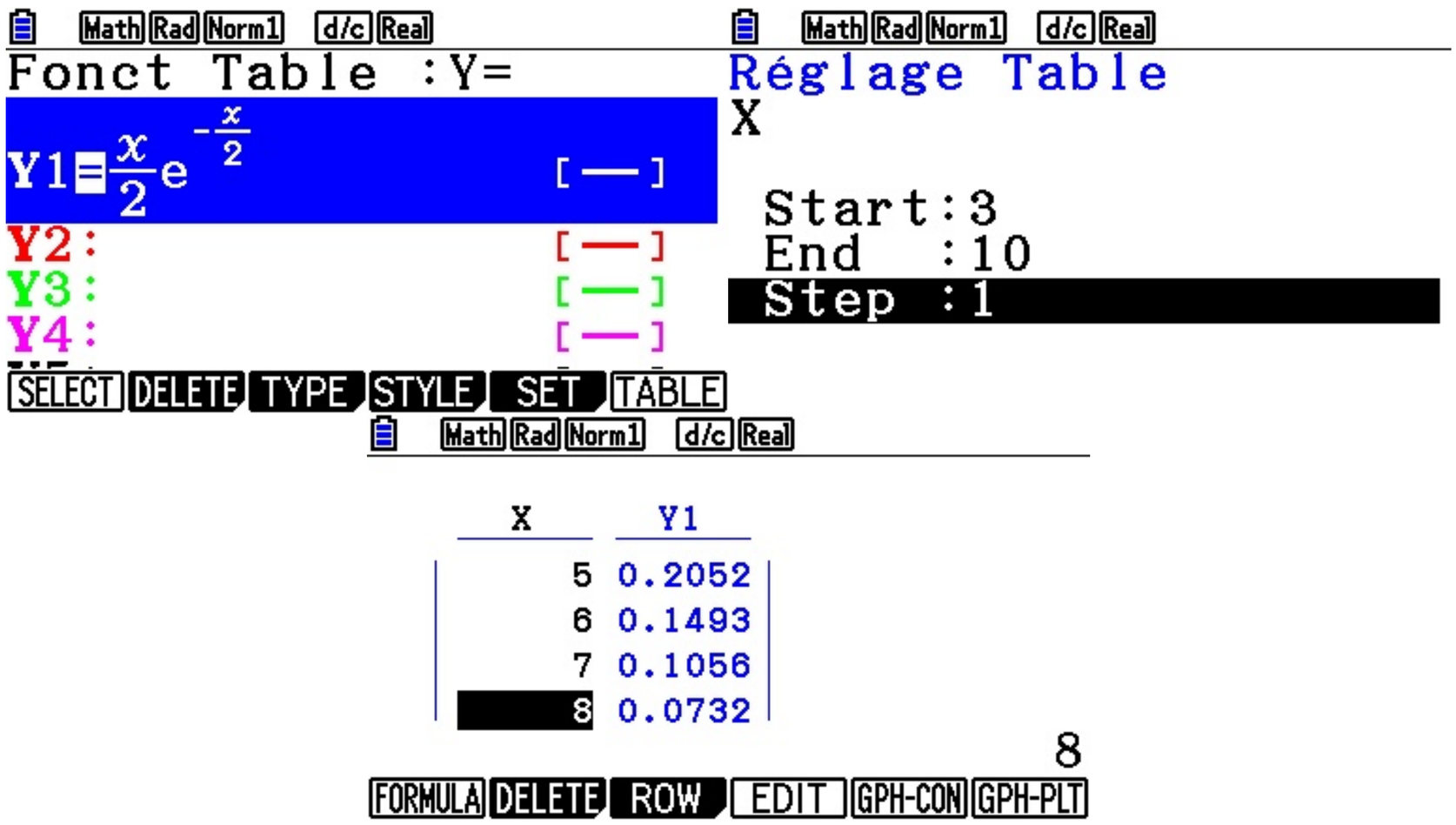
Question B)3)b):
On nous demande maintenant ce que répond l'algorithme.
Si on bien compris ce qu'il faisait, on peut se contenter de demander tout simplement à la calculatrice un tableau de valeurs pour la fonction f.

Sur TI-82 à TI-84, il suffit d'abord de saisir la fonction avec STAT PLOT f(x) Y=, puis de configurer le tableau de valeurs avec 2nde 2ND fenêtre WINDOW (à partir de 3 et avec un pas de 1 ici donc) et enfin d'en demander l'affichage avec 2nde 2ND graphe GRAPH :

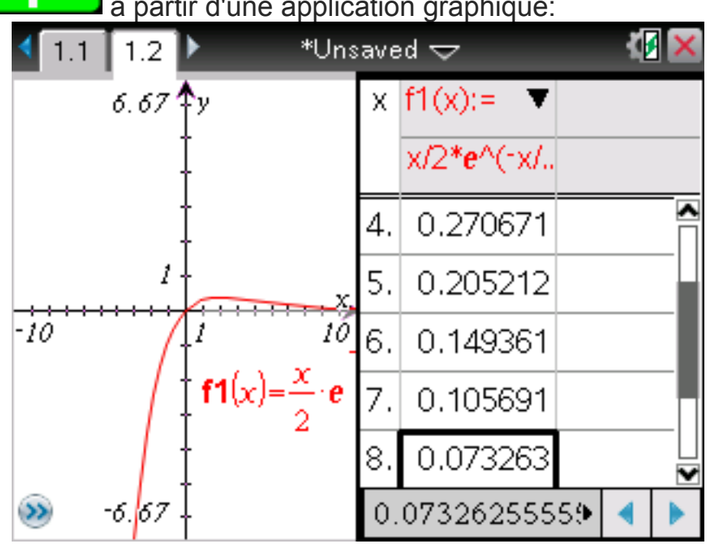


La première valeur de n vérifiant $f(n)\leq 0,1$ est donc $n_0=8$.

La même chose est réalisable sur Casio Graph et Casio Prizm:



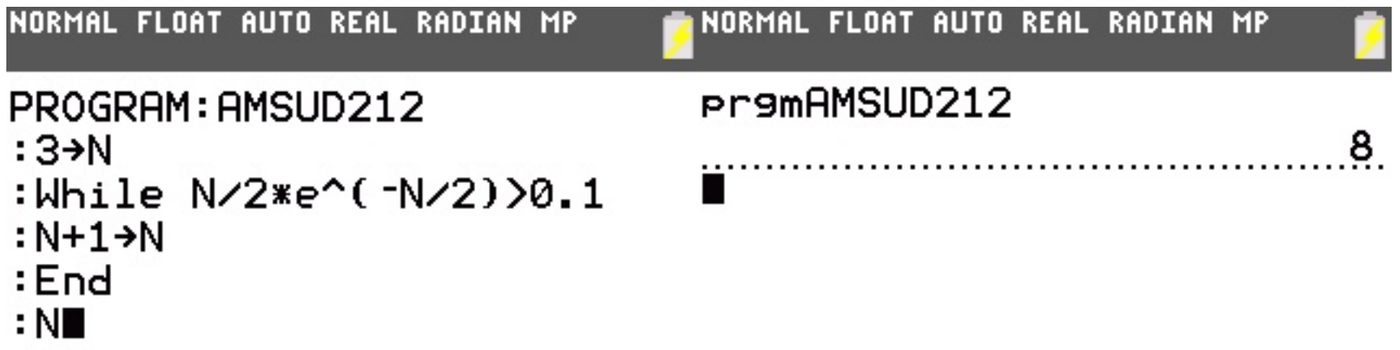
Ou encore sur TI-Nspire en utilisant le raccourci ctrl T à partir d'une application graphique:



Dans tous les cas on trouve bien évidemment n_0=8.

Le seul ennui de cette méthode, est que si l'on a mal compris l'algorithme on se trompe...
La méthode sans danger qui demandera à peine plus de temps si vous vous êtes bien entraîné, est de programmer tout simplement l'algorithme sur votre calculatrice et là le résultat sera fiable!

Voici donc là encore la solution pour les principaux modèles de calculatrices graphiques:



Question B)3)c):
On nous demande enfin d'interpréter ce résultat.
Il s'agit donc du nombre d'heures au bout duquel la quantité de principe actif présent dans le sang chutera en dessous de 0,1mg.

A bientôt!

Pack et mise en page par Laurae, rédigé par critor sur le forum, administrateurs de TI-Planet.org

Correction 18 : Correction algorithme Concours Général Mathématiques 2013

de **critor** » 30 Avr 2013 14:19

L'épreuve de mathématiques du BAC S a suivi nombre de modes depuis bientôt 10 ans que je la suis.

Il y a eu la mode QCM (2004-2005), puis la mode ROC (2006), puis la mode "question ouverte"...

Nous sommes clairement aujourd'hui dans la mode algorithmique, d'une popularité sans précédent puisque s'étendant même au delà de la série S (séries ES, L et bientôt technologiques), et même au-delà du BAC!

En effet, un exercice d'algorithmique vient de tomber au Concours Général de Mathématiques 2013.
Le voici:

PROBLÈME III

Il faut passer les premiers

Pour ce problème, on donne la liste des vingt-cinq nombres premiers inférieurs à 100 :

$$2,3,5,7,11,13,17,19,23,29,31,37,41,43,47,53,59,61,67,71,73,79,83,89,97.$$

Pour faire une réussite, Sisyphe a dessiné au sol 106 cases numérotées de 0 à 105, et il dispose d'un jeton et d'un dé à six faces (équilibré).
Sisyphe commence la réussite en posant le jeton sur la case 0. Il fait ensuite une série de lancers du dé ; lorsque le dé affiche la valeur k, il avance le jeton de k cases et :
– s'il atteint ou dépasse la case numéro 100, Sisyphe a gagné ;
– s'il arrive à une case dont le numéro est un nombre premier inférieur à 100, Sisyphe a perdu ;
– dans les autres cas, Sisyphe relance le dé et continue la réussite.

1° Dans cette question, on suppose que Sisyphe recommence une réussite lorsqu'il a perdu.
On note p_n la probabilité de gagner au moins une réussite en au plus n lancers du dé.
a) Déterminer les valeurs de n pour lesquelles $p_n > 0$.
b) Étudier la convergence de la suite (p_n).

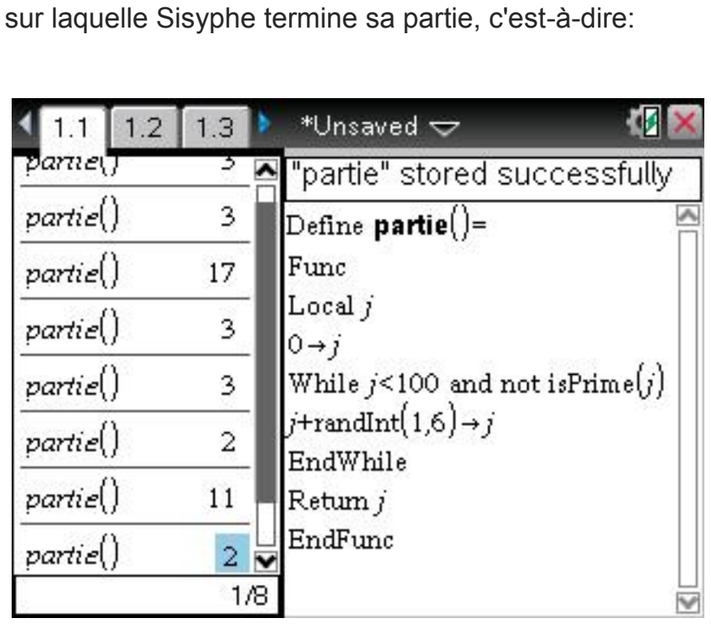
Dans la suite du problème, Sisyphe ne recommence plus la réussite s'il perd.
Soit X la variable aléatoire représentant la position du jeton à la fin de la réussite.
On note $P(X = k)$ la probabilité de l'événement $X = k$.

2° a) Déterminer $P(X = 2)$, $P(X = 3)$, $P(X \geqslant 4)$, $P(X = 5)$.
b) Proposer un algorithme pour calculer $P(X = k)$ pour tout $k \in \{0,\ldots,105\}$.

3° Muni d'une calculatrice qui n'est pas assez performante pour exécuter l'algorithme précédent, Sisyphe cherche à estimer sa probabilité de gain. Pour cela, étant donné deux nombres premiers consécutifs $p < p'$, il considère α_p la probabilité conditionnelle de l'événement $X = p'$ sachant l'événement $X > p$.
a) Que valent α_2 et α_3 ?
b) Donner l'expression de la probabilité de gain, $P(G) = P(X \geqslant 100)$, en fonction des nombres réels α_p pour $p = 2, 3, 5,\ldots$
c) Donner un encadrement des nombres α_p et en déduire un encadrement de $P(G)$. (Dans cette question, la qualité de l'encadrement sera un élément d'appréciation.)

Nous nous devons donc d'aider Sisyphe, qui avance à partir de la case '0' avec un dé à 6 faces, et se doit d'atteindre ou dépasser 100 tout en évitant les nombres premiers.

Mais il n'y a que 25 nombres premiers inférieurs à 100 - un jeu d'enfant me direz-vous?
Approchons le problème de façon empirique à l'aide d'une petite simulation.
Voici un programme TI-Nspire qui renvoi le numéro de la case sur laquelle Sisyphe termine sa partie, c'est-à-dire:
- un nombre supérieur ou égal à 100 si il gagne
- un nombre premier inférieur à 100 si il perd

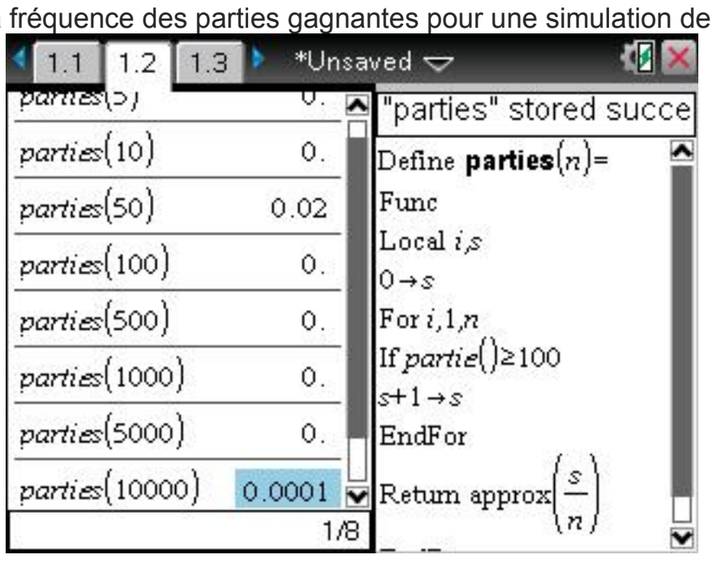


Comme vous voyez, j'ai effectué 8 parties et ai toujours perdu en atteignant un nombre premier inférieur à 100...

Mais peut-être est-ce de la malchance? Il faut simuler un grand nombre de parties pour que la fréquence d'apparition d'un événement tende vers sa probabilité.

Pack et mise en page par Laurae, rédigé par critor sur le forum, administrateurs de TI-Planet.org

Voici un autre programme TI-Nspire destiné à nous renvoyer la fréquence des parties gagnantes pour une simulation de n parties:

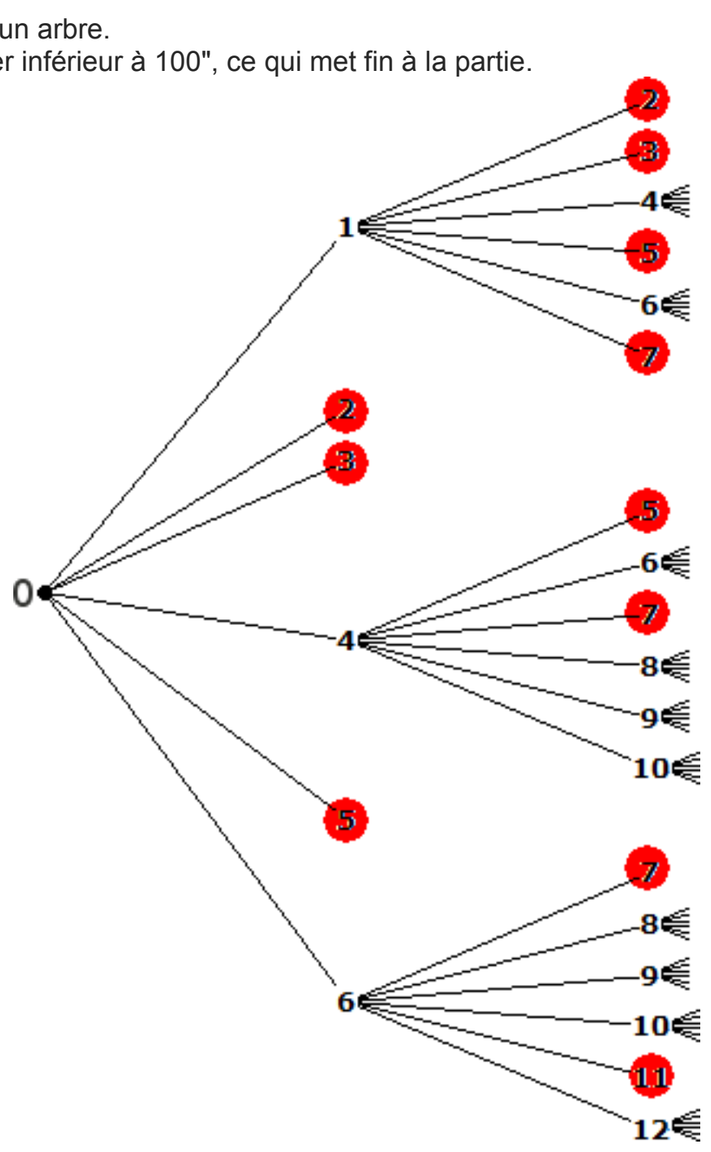


La probabilité de gagner la partie est effectivement très faible... Malgré plusieurs centaines ou milliers de parties simulées, la fréquence des parties gagnantes en souvent nulle ou très proche de zéro...

Les apparences sont donc trompeuses, et ce pauvre Sisyphe porte donc bien son nom, tel son homologue mythologique père d'Ulysse, qui fut damné à pousser un rocher jusqu'en haut d'une colline, ce rocher ayant une très forte probabilité de redégringoler alors immédiatement puisqu'il s'agit d'un équilibre instable (dérivée seconde du potentiel négative).

Bon, venons-en enfin aux questions qui nous intéressent.
Dans la 2ème partie, on étudie la loi de probabilité de variable aléatoire X, où X prend pour valeurs le numéro de la case en fin de partie (qu'elle soit gagnée ou perdue).

Les événements X=2 et X=3 sont aisément dénombrables via un arbre.
On code ici en rouge l'événement "atteindre un nombre premier inférieur à 100", ce qui met fin à la partie.



A partir de la case 0, on peut atteindre la case 2 en:
- sortant directement 2 (probabilité 1/6)
- sortant 1 puis 1 (probabilité 1/36)

Donc P(X=2)=7/36.

La case 2 étant interdite (nombre premier), les seules façons d'atteindre 3 sont en:
- sortant directement 3 (probabilité 1/6)
- sortant 1 puis 2 (case intermédiaire: 1 - probabilité 1/36)

Donc encore P(X=3)=7/36.

Les cases 2 et 3 étant interdites (nombres premiers), les seules façons d'atteindre 5 sont en:
- sortant directement 5 (probabilité 1/6)
- sortant 4 puis 1 (case intermédiaire: 4 - probabilité 1/36)
- sortant 1 puis 4 (case intermédiaire: 1 - probabilité 1/36)
- sortant 1 puis 3 puis 1 (cases intermédiaires: 1, 4 - probabilité 1/216)

Donc P(X=5)=49/216.

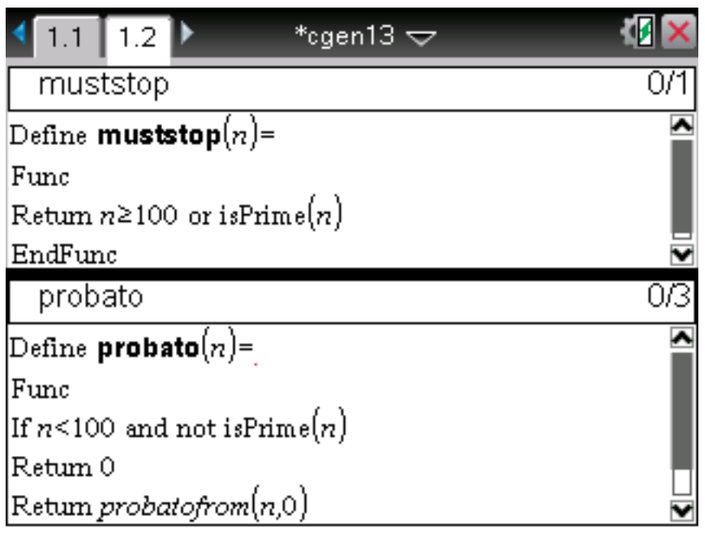
Les nombres 0, 1 et 4 étant des nombres non premiers ils ne peuvent achever une partie et on a donc de façon triviale P(X=0)=P(X=1)=P(X=4)=0.

On en déduit que P(X>=4)=1-P(X<4)=1-(P(X=0)+P(X=1)+P(X=2)+P(X=3))=11/18

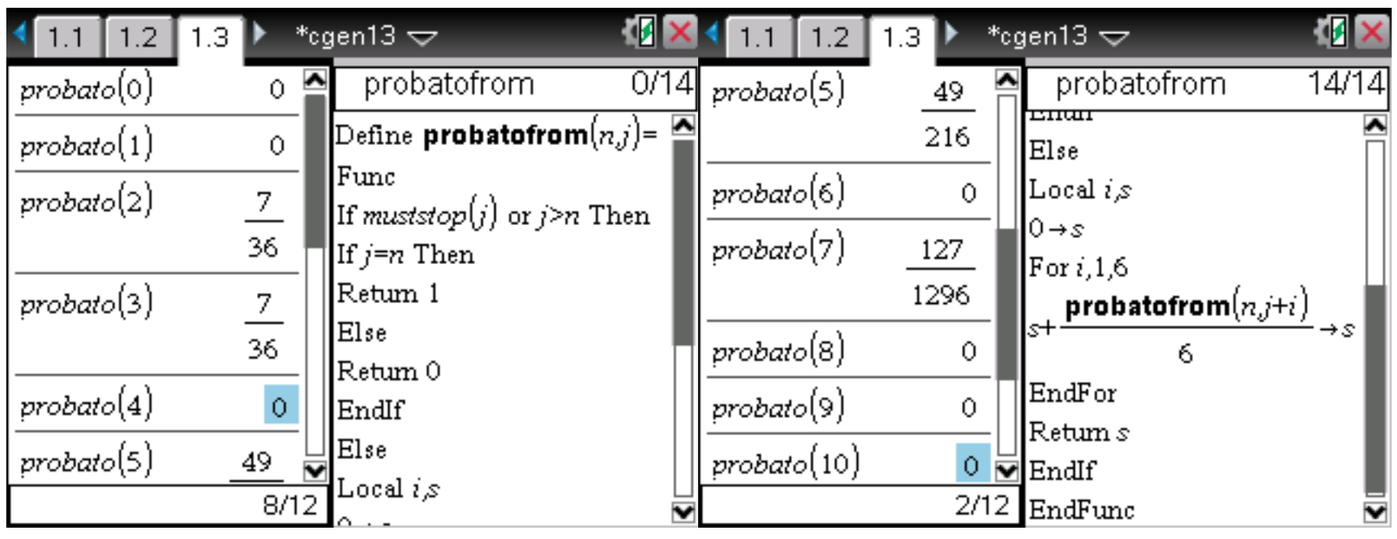
Mais nous aurons bien évidemment beaucoup de mal à esquisser un arbre qui va plus loin...

La question nous demande donc en conséquence la production d'un algorithme permettant de déterminer P(X=k).
En utilisant les formules de probabilités d'événements successifs indépendants, on peut construire un algorithme récursif qui va parcourir l'arbre en profondeur, chaque nouvelle sous-branche correspondant à une multiplication par 1/6.

Nous allons utiliser pour cela sur TI-Nspire:
- une fonction testant si le noeud de l'arbre est un noeud arrêtant la partie
- une fonction pour initialiser la récursivité



Et il nous faut enfin la fonction récursive en tant que telle, qui à partir d'un noeud-parent se réappelle sur les 6 noeuds-fils (dé tiré entre 1 et 6) pour calculer les probabilités par multiplication par 1/6 et somme.
Cette fonction travaille donc branche par branche.
Elle renverra 0 si on termine la partie sans atteindre le nombre cherché, ce qui par produit annulera les chemins inutiles ici et permettra de ne considérer par somme que les 'bons' chemins.

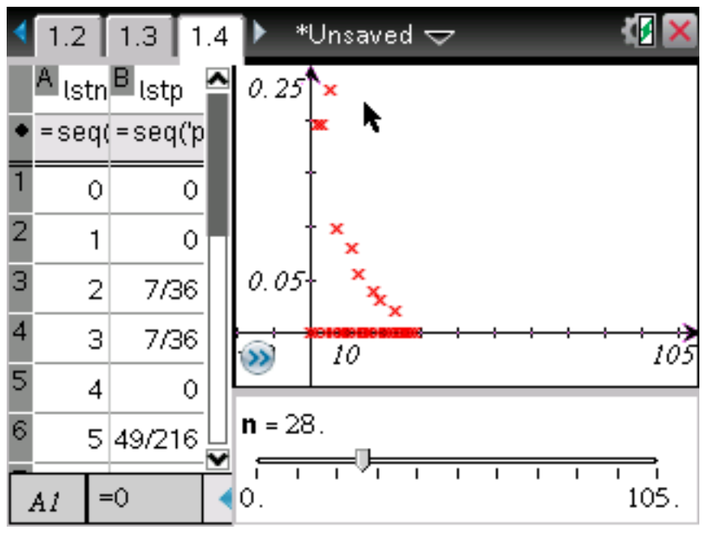


Il ne s'agit probablement pas de l'algorithme le plus rapide, et une version itérative serait sans doute plus performante.
Mais je trouvais la version récursive plus naturelle à comprendre.

Dans tous les cas, il s'agit probablement d'un problème de complexité exponentielle, et quoi que l'on fasse l'algorithme explosera donc rapidement les capacités de toutes les machines actuelles.

Sur TI-Nspire notamment, en partant de P(X=0) on passe de calculs instantanés à des calculs de quelques secondes, puis quelques minutes, et enfin quelques heures sur P(X=29).

Pack et mise en page par Laurae, rédigé par critor sur le forum, administrateurs de TI-Planet.org



Tout juste peut-on conjecturer que la suite des P(X=k) non nuls semble décroissante à partir du rang 5 et tendre rapidement vers 0.
D'ici à ce qu'elle arrive à P(X=105), l'univers a certainement le temps de s'écrouler...

Edit: Pour une version itérative de l'algorithme, voir les commentaires de la news.

Pack et mise en page par Laurae, rédigé par critor sur le forum, administrateurs de TI-Planet.org

Correction 19 : Correction algo Olympiades Académiques 2013 1S Aix-Marseille

de **critor** » 02 Mai 2013 15:25

Après le BAC et le Concours Général, aux Olympiades Académiques 2013 de 1ère sont tombés de nombreux algorithmes.

Intéressons-nous aujourd'hui à l'algorithmique qui est tombé en exercice 3 pour les Premières S dans l'Académie d'Aix-Marseille:

1. On considère l'équation :

$$81a+125b+149c=2013 \quad (E)$$

où a, b et c sont des entiers positifs.

a) Justifier que a est nécessairement compris entre 0 et 24.
Donner un encadrement possible des valeurs prises respectivement par b puis par c.

b) Écrire un algorithme permettant :

- de tester chacune des valeurs a, b et c prises dans les encadrements précédemment définis afin de vérifier si $81a+125b+149c=2013$;
- d'afficher les valeurs a, b et c, si elles existent, solutions de (E).

c) On fixe $a=15$. En utilisant la méthode de votre choix, déterminer tous les triplets $(15\,;\,b\,;\,c)$ de nombres entiers positifs, s'il en existe, solutions de (E).

Question 1)a):
Nous étudions donc l'équation (E) 81a+125b+149c=2013, où a, b et c sont des entiers naturels.

Nous avons donc b≥0 et c≥0.
On en déduit 125b≥0 et 149c≥0.
Par sommation des inégalités, 125b+149c≥0

Mais l'équation (E) peut aussi s'écrire 125b+149c=2013-81a.
On en déduit 2013-81a≥0,

D'où: 2013≥81a
2013/81≥a
a≤2013/81

Comme a est un entier positif et que 2013/81≈24,9 on en déduit que 0≤a≤24.

On montre de même que b≤2013/125 et c≤2013/149, ce qui donne 0≤b≤16 et 0≤c≤13.

Question 1)b):
On nous demande donc maintenant d'écrire un algorithme recherchant tous les triplets (a,b,c) solutions de (E).

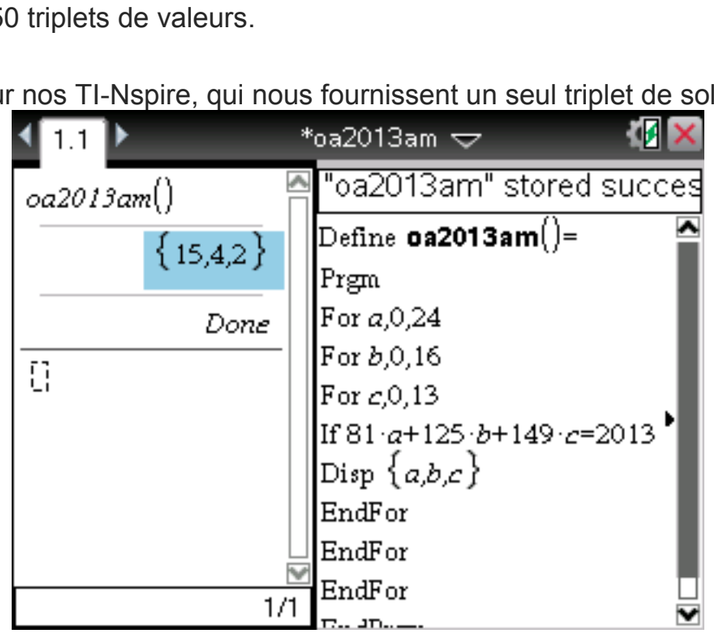
Il s'agit donc de vérifier l'équation (E) pour tous les triplets de valeurs possibles pour (a,b,c).

L'on peut faire cela en imbriquant 3 boucles 'pour':

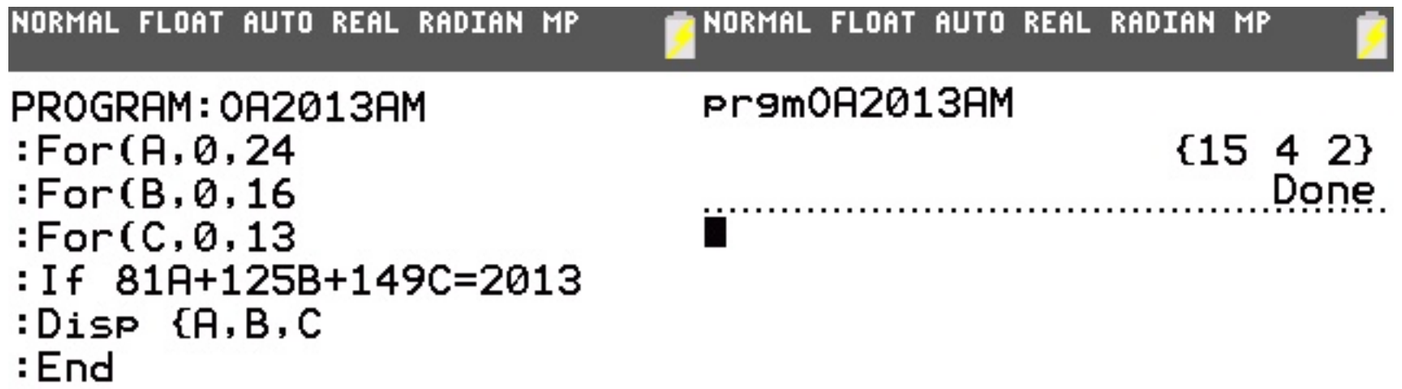
```
Pour a de 0 à 24 faire
   Pour b de 0 à 16 faire
      Pour c de 0 à 13 faire
         Si 81a+125b+149c=2013 alors
            Afficher (a,b,c)
         FinSi
      FinPour
   FinPour
FinPour
```

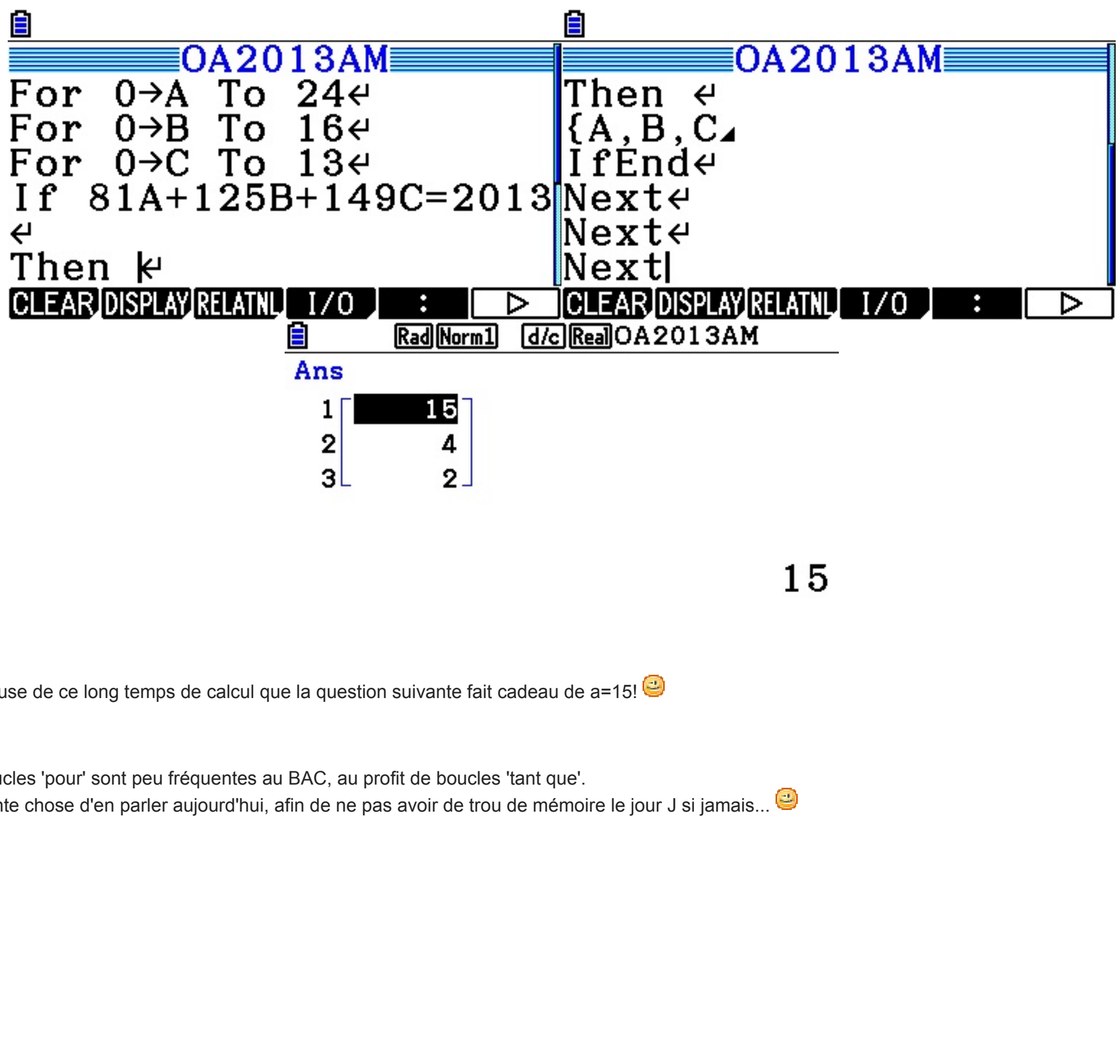
Remarquons que cet algorithme revient à tester 25*17*14=5950 triplets de valeurs.

Il est possible de traduire cet algorithme en un programme pour nos TI-Nspire, qui nous fournissent un seul triplet de solutions (15,4,2) en seulement quelques secondes!



Le même programme est réalisable sur nos TI-82 à TI-84 ou Casio Graph/Prizm, mais il faudra cette fois-ci patienter quelques dizaines de secondes avant d'obtenir les mêmes solutions:





Sans doute est-ce à cause de ce long temps de calcul que la question suivante fait cadeau de a=15!

Jusqu'à présent les boucles 'pour' sont peu fréquentes au BAC, au profit de boucles 'tant que'.
C'est donc une excellente chose d'en parler aujourd'hui, afin de ne pas avoir de trou de mémoire le jour J si jamais...

Pack et mise en page par Laurae, rédigé par critor sur le forum, administrateurs de TI-Planet.org

Correction 20 : Correction algo Olympiades Académiques 2013 1ère Besançon

de **critor** » 02 Mai 2013 18:05

Après l'algorithme des Olympiades Académiques 2013 d'Aix-Marseille dans une news précédente, intéressons-nous maintenant à l'algorithme tombé dans l'Académie de Besançon.

EXERCICE 3 *(académique)*

On définit la fonction σ sur l'ensemble $\mathbb{N}^*$ des entiers naturels non nuls, qui à n associe la somme de ses diviseurs.

Exemple : $\sigma(21) = 32$ car les diviseurs de 21 sont 1 ; 3 ; 7 ; 21 et $1 + 3 + 7 + 21 = 32$.

Définition : Un entier naturel p est dit *premier* s'il admet deux diviseurs distincts : 1 et p.

Théorème 1 (*Décomposition en facteurs premiers*)

Tout nombre entier naturel supérieur ou égal à 2 peut s'écrire comme un produit de nombres premiers (non nécessairement distincts).

Exemple : $72 = 2^3 \times 3^2$.

On rappelle également les deux propriétés suivantes :

Propriété 1

Pour tout entier naturel non nul n,

$$1 + 2 + \ldots + n = \frac{n(n+1)}{2}.$$

Propriété 2

Pour tout nombre réel $x \neq 1$ et pour tout entier naturel n,

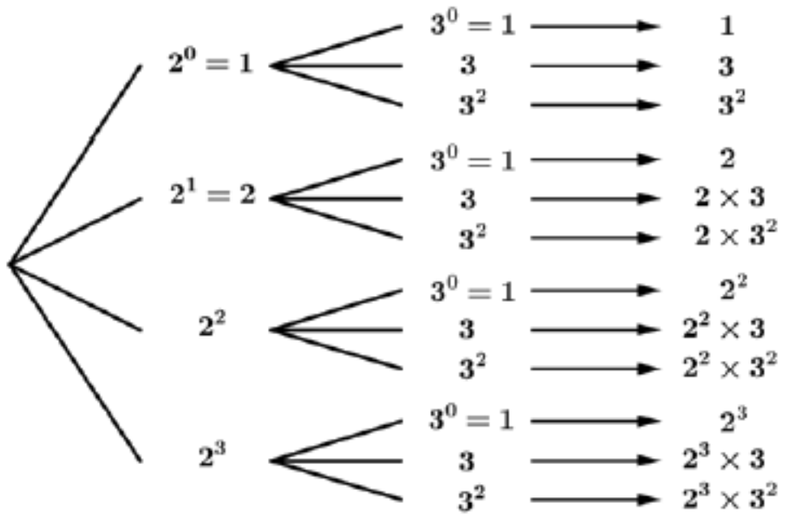
$$1 + x + x^2 + \ldots + x^n = \frac{x^{n+1} - 1}{x - 1}.$$

Partie I : Obtention des diviseurs d'un nombre entier naturel

La décomposition en facteurs premiers d'un entier naturel permet d'obtenir tous ses diviseurs de manière systématique.
On a vu par exemple que $72 = 2^3 \times 3^2$. Les diviseurs de 72 sont alors :

$$1\ ;\ 2\ ;\ 2^2\ ;\ 2^3\ ;\ 3\ ;\ 3^2\ ;\ 2\times 3\ ;\ 2^2\times 3\ ;\ 2^3\times 3\ ;\ 2\times 3^2\ ;\ 2^2\times 3^2\ ;\ 2^3\times 3^2.$$

On peut s'aider d'un arbre pour lister ces diviseurs :



1. Donner la décomposition en facteurs premiers de 350 et en déduire la liste de ses diviseurs. Calculer $\sigma(350)$.
2. On considère l'algorithme incomplet suivant dont le rôle est de calculer $\sigma(n)$ connaissant n :

Entrée	:	n entier naturel non nul
Initialisation	:	σ prend la valeur 0

Traitement	:	Pour k allant de ... à ... faire :
		  Si le reste de la division euclidienne de n par k est 0, alors :
		  Affecter à σ la valeur ...
		  Fin Si
		Fin Pour

Sortie	:	Afficher σ

Recopier et compléter cet algorithme en remplaçant la condition « le reste dans la division euclidienne de n par k est 0 » par une instruction mathématique.

Partie II : Quelques propriétés de σ

1. Déterminer $\sigma(n)$ pour $n = 1$, $n = 2$, $n = 3$, $n = 4$, $n = 5$ et $n = 6$.

Question I)2):
On nous donne donc un algorithme à trous, destiné à calculer σ(n) pour tout entier naturel n non nul, où σ(n) est la somme de tous les diviseurs de n.

Lorsque le test "le reste de la division euclidienne de n par k est 0" est vérifié, cela veut dire que k est un diviseur de n.
Il faut donc l'ajouter à la somme des diviseurs déjà trouvée.
Il nous faut donc une variable pour cumuler les diviseurs trouvés au fur et à mesure, et c'est la variable σ initialisée à 0, élément neutre de l'addition, qui joue ici ce rôle.
(si on avait du multiplier les valeurs trouvées au lieu de les additionner, on aurait initialisé la variable à 1, élément neutre de la multiplication)
La dernière instruction manquante sera donc "Affecter à σ la valeur σ+k".

Ce test peut être écrit mathématiquement en utilisant la fonction partie entière E introduite en début de Première S.
On peut alors le récrire par exemple "E(n/k)=n/k".

Enfin, k jouant le rôle des diviseurs de n recherchés, on a 1≤k≤n, ce qui nous permet de compléter les bornes de la boucle 'pour':
"Pour k allant de 1 à n faire"

L'algorithme une fois complété nous donne donc:
CODE: TOUT SÉLECTIONNER

```
Entrée:
   n entier naturel non nul
Initialisation:
   σ prend la valeur 0
Traitement:
   Pour k allant de 1 à n faire
      Si E(n/k)=n/k alors
         Affecter à σ la valeur σ+k
      FinSi
   FinPour
   Afficher σ
```

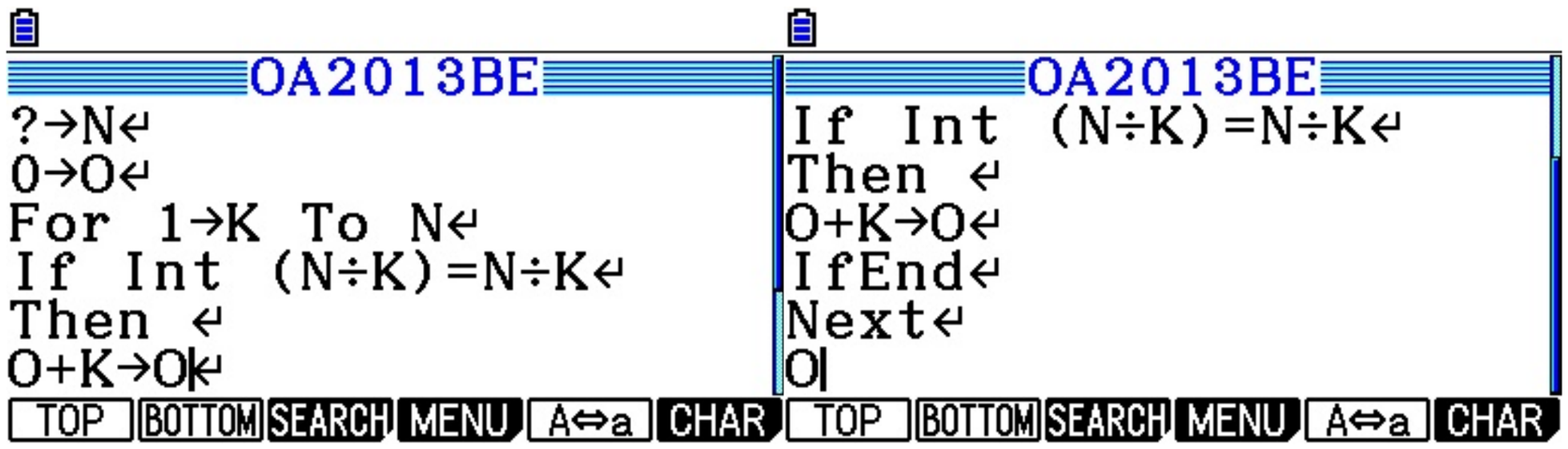
L'on traduit aisément l'algorithme en un programme pour nos TI-82 à TI-84:

```
NORMAL FLOAT AUTO REAL RADIAN MP
PROGRAM:0A2013BE
:Prompt N
:0→O
:For(K,1,N
:If int(N/K)=N/K
:Then
:O+K→O
:End
:End
:O
```

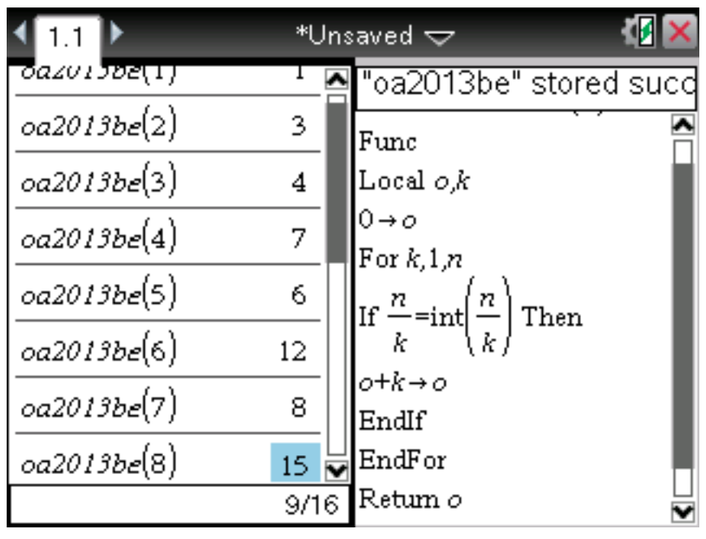
Il est alors aisé de vérifier le résultat de la question I)1) précédente: σ(350)=744

```
NORMAL FLOAT AUTO REAL RADIAN MP
prgm0A2013BE
N=?350
                                744
```

Voici le programme pour Casio Graph/Prizm:



Et voici maintenant le programme pour TI-Nspire:



Une fois le programme fonctionnel, il est alors aisé et rapide de déterminer quelques valeurs de σ(n), ce qui va être utile pour les quelques exemples demandés par les questions suivantes!

Question II)1)

Il nous faut donc déterminer justement σ(n) pour n allant de 1 à 6.
Le programme nous répond rapidement que σ(1)=1, σ(2)=3, σ(3)=4, σ(4)=7, σ(5)=6 et σ(6)=12.

Partie III : Nombres parfaits

Définition : Un entier naturel non nul n est dit *parfait* s'il est égal à la somme de ses diviseurs autres que lui-même.

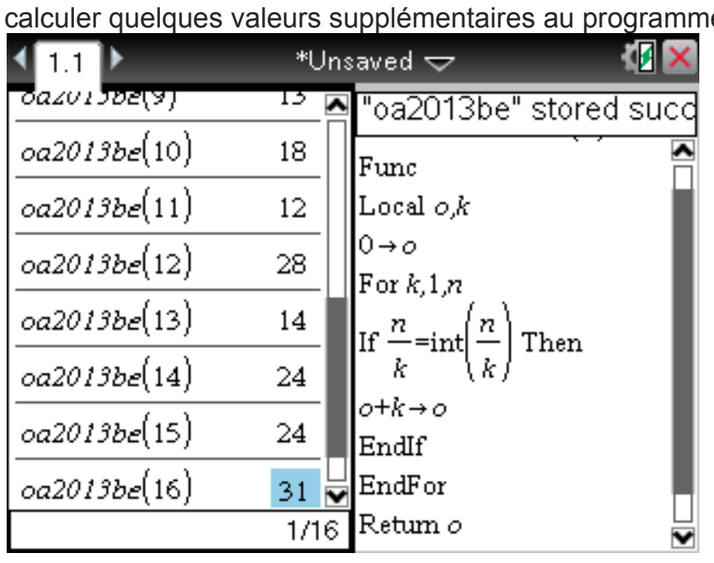
Exemple : 28 est un nombre parfait car les diviseurs de 28, différents de 28, sont 1 ; 2 ; 4 ; 7 ; 14 et on a :

$$28 = 1 + 2 + 4 + 7 + 14$$

1. Justifier que si n est un entier naturel parfait, alors $\sigma(n) = 2n$.
2. Existe-t-il des nombres premiers parfaits ?
3. a) Déterminer tous les nombres parfaits inférieurs ou égaux à 10.

Question III)3)a)

Sachant qu'un nombre parfait n vérifie σ(n)=2n, il suffit de faire calculer quelques valeurs supplémentaires au programme:



On vérifie alors aisément dans la liste des résultats qu'avec σ(6)=12, 6 est le seul nombre parfait inférieur ou égal à 10.

5. Un entier naturel non nul n est dit *presque parfait* si $\sigma(n) = 2n - 1$.
   a) Déterminer tous les nombres presque parfaits inférieurs ou égaux à 16.

Pack et mise en page par Laurae, rédigé par critor sur le forum, administrateurs de TI-Planet.org

Question III)5)a)

Sachant qu'un nombre presque parfait n vérifie σ(n)=2n-1, on obtient rapidement de façon similaire que les seuls nombres presque parfaits inférieurs ou égaux à 16 sont 1, 2, 4, 8 et 16 avec σ(1)=1, σ(2)=3, σ(4)=7, σ(8)=15 et σ(16)=31.

Tiens, ne seraient-ce pas les puissances de 2?...

Pack et mise en page par Laurae, rédigé par critor sur le forum, administrateurs de TI-Planet.org

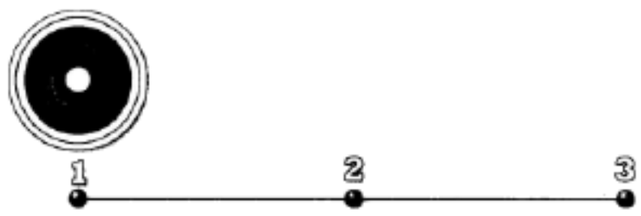
Correction 21 : Correction algo Olympiades Académiques 2013 1èreS Mayotte

de **critor** » 03 Mai 2013 00:46

Après les algorithmes des Académies d'Aix-Marseille et Besançon dans deux news précédentes, regardons ce soir ensemble l'algorithme tombé en exercice 4 aux Olympiades Académiques de Mathématiques de Première S 2013, dans l'Académie de Mayotte cette fois-ci.

Exercice 4 : marche aléatoire **A traiter uniquement par les candidats de S**

Partie A : Marche aléatoire sur un segment



Dans un jeu vidéo, une cible se déplace sur un segment de la manière suivante :
Elle part de la position 1 puis change de position toutes les secondes en suivant les règles ci-dessous :

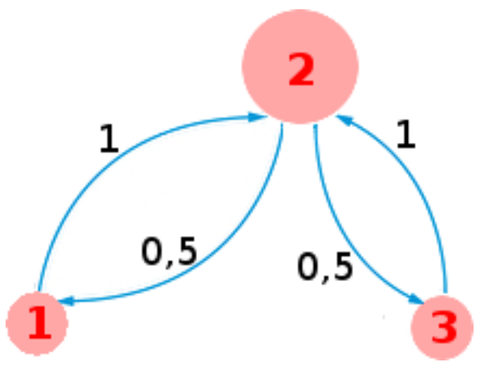
- Des positions 1 et 3, elle se déplace à la position 2 ;
- De la position 2, elle se déplace soit à la position 1 soit à la position 3, avec des probabilités identiques.

1. Quelle est la probabilité que la cible soit à la position 1, trente secondes plus tard.
2. Soit n un entier naturel impair.
   a. Déterminer la probabilité que la cible soit en position 1 au bout de n secondes.
   b. Déterminer la probabilité que la cible soit en position 2 au bout de n secondes.
3. Soit n un entier naturel non nul et pair.
   a. Déterminer la probabilité que la cible soit en position 1 au bout de n secondes.
   b. Déterminer la probabilité que la cible soit en position 2 au bout de n secondes.

Il s'agit donc d'étudier les déplacements d'une cible chaque seconde entre trois positions 1, 2, 3 selon les règles suivantes:

- la cible commence en 1
- de 1 et de 3, la cible va en 2
- de 2, la cible va en 1 ou 3 de façon équiprobable

L'on peut représenter cette situation à l'aide d'un graphe probabiliste:



Selon le graphe en partant de 1, après un nombre impair de secondes/déplacements, on est forcément en 2.
Après un nombre pair non nul de déplacements, on se retrouve de façon équiprobable, soit en 1, soit en 3.

Question A)1) 1/2
Question A)2)a) 0
Question A)2)b) 1
Question A)3)a) 1/2
Question A)3)b) 0

Question A)4) 0

4. Compléter les deux algorithmes « à trous » ci-dessous de manière à ce que chacun d'entre eux permette de simuler la position de la cible au bout de n secondes.

   La fonction « Entalea(0 ;1) » permet l'affichage, de manière aléatoire mais équiprobable, de l'entier 0 ou de l'entier 1.

Algorithme n°1

```
Variables : n entier
Début
        Entrer (n) ;
        Si n est impair
                Alors Afficher (« ..... ») ;
        FinSi ;
        Si n est pair
                Alors Si Entalea(0 ;1) = 0
                                Alors Afficher (« ..... ») ;
                                Sinon Afficher (« ..... ») ;
                        FinSi ;
        FinSi ;
Fin
```

Notons que cet algorithme est fort mal écrit - avec une fonction EntAlea() qui n'est ni du langage naturel ni du langage mathématique, des parenthèses pour des affichages et des points virgule de séparation de paramètres ou de ponctuation d'instructions.

Un excellent exemple de ce qu'il ne faut pas faire au BAC!
Cela ressemble fortement à un langage de programmation car il y a des contraintes de syntaxe et l'auteur en aurait donc rapidement traduit les instructions en français, ce qui justement n'est **pas** un **algorithme**.

Les trous à compléter dans l'algorithme correspondent simplement aux cas étudiés ci-dessus.
On peut donc les compléter de la façon suivante:

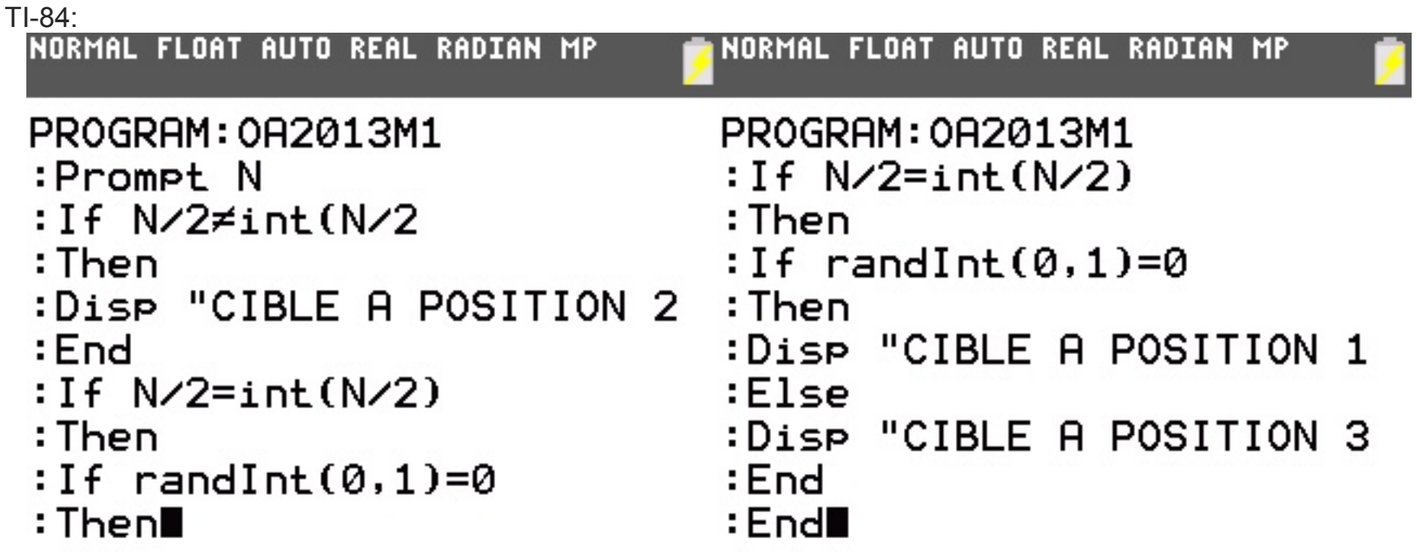
CODE: TOUT SÉLECTIONNER

```
Variables: n entier
Début
   Entrer n
   Si n est impair alors
      Afficher "Cible à position 2"
   sinon
      Si EntAlea(0,1)=0 alors
         Afficher "Cible à position 1"
      sinon
         Afficher "Cible à position 3"
      FinSi
   FinSi
Fin
```

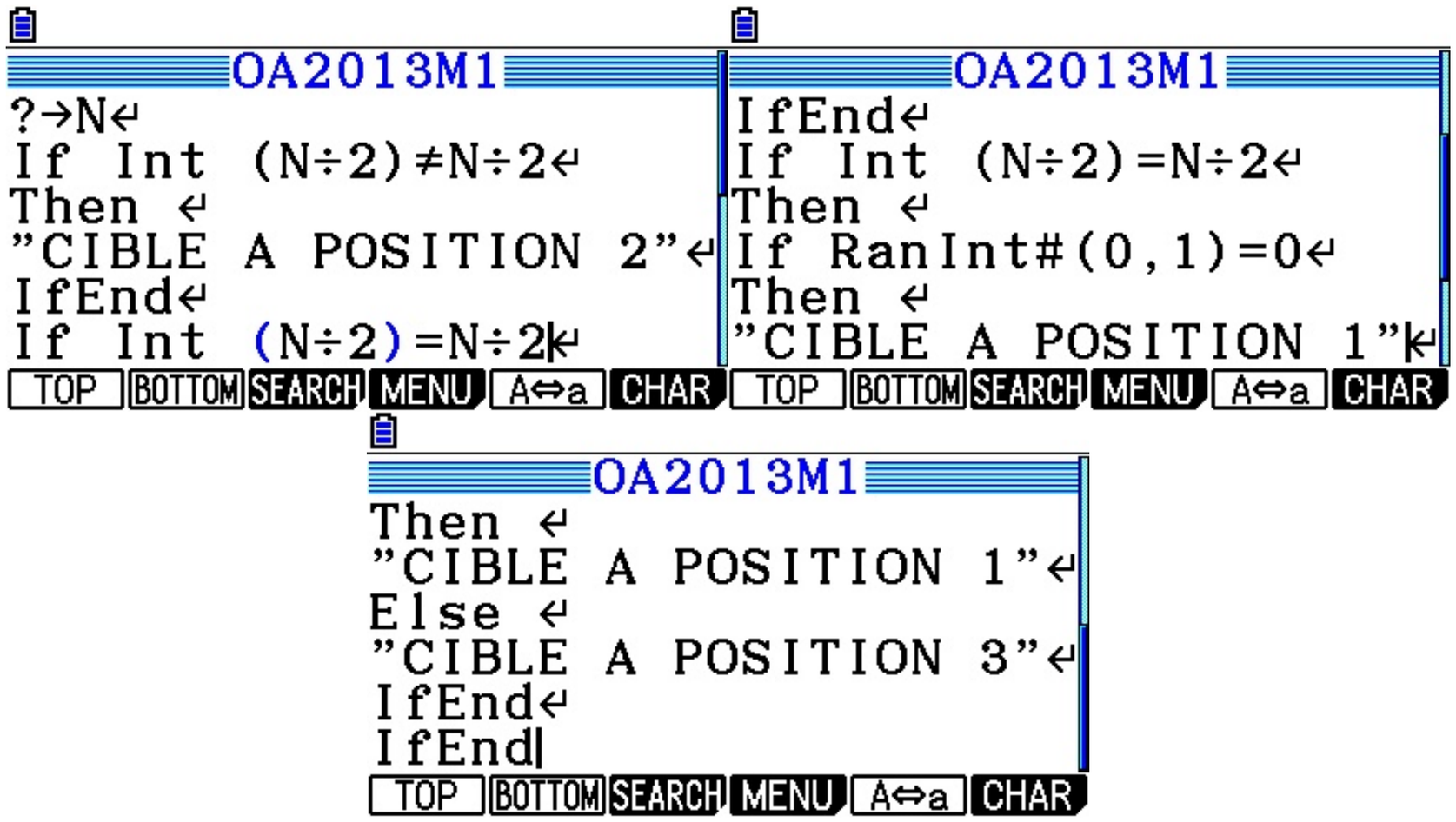
L'on vérifie aisément le fonctionnement correct de l'algorithme en le traduisant en un programme pour notre calculatrice.

Le test de parité de N peut utiliser la fonction partie entière afin de vérifier si N est divisible par 2 ou non.

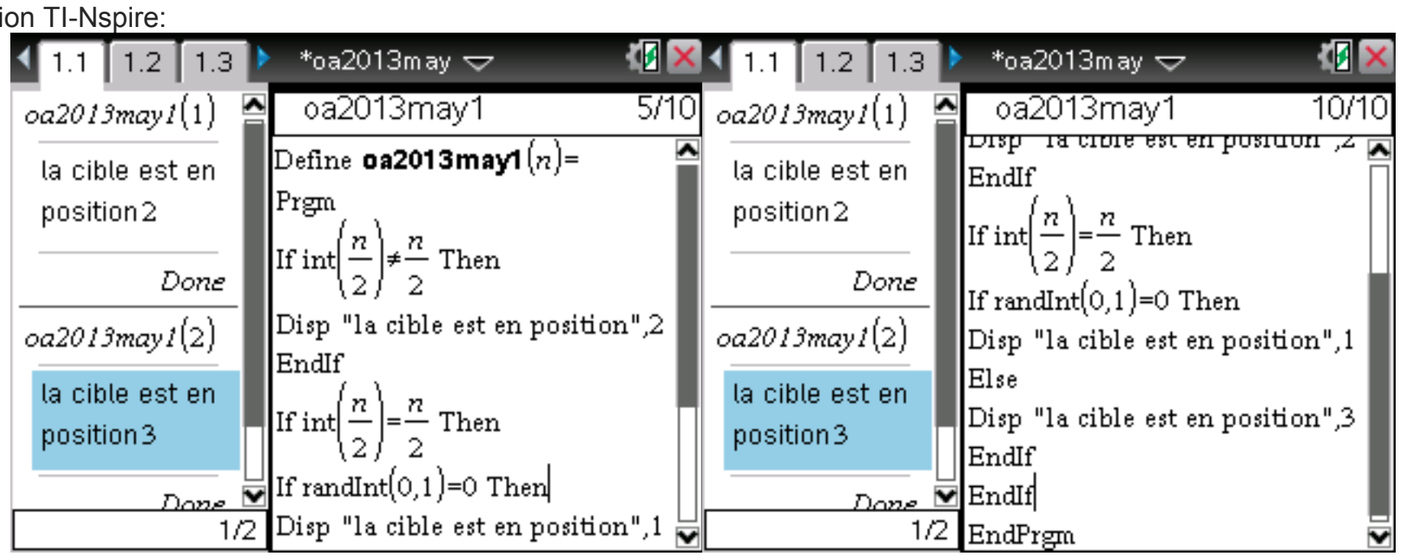
Voici un programme pour TI-82 à TI-84:



En voici maintenant un pour Casio Graph/Prizm:



Et enfin maintenant, voici une version TI-Nspire:



Algorithme n°2 :

```
Variables : n, C, i entiers
Début
   Entrer(n) ;
   C←1 ;
   Pour i = …. à n Faire
            Si ……..
                     Alors C← 2 ;
                          Sinon  Si Entalea(0 ;1) = 0
                                              Alors C← 1 ;
                                              Sinon C←….. ;
            FinSi ;
   FinPour ;
   Afficher (« la cible est en position »,C) ;
Fin
```

Encore un 'algorithme' assez mal écrit pour les mêmes raisons que le précédent, d'autant plus qu'il y a deux instructions 'Si' mais un seul 'FinSi'.

Cette fois-ci on utilise une boucle "pour i=... à n faire".
n étant le nombre de déplacements de la cible, nous allons mettre "pour i=1 à n faire" afin de passer exactement n fois dans la boucle.

La cible va en 2 lorsque qu'elle est en 1 ou 3.
Nous complétons donc l'instruction conditionnelle avec "Si C=2 ou C=3".

Enfin, nous avons des affectations de C avec 2 et 1, la dernière affectation correspond donc forcément par élimination à 3, et nous mettons "Sinon C←3".

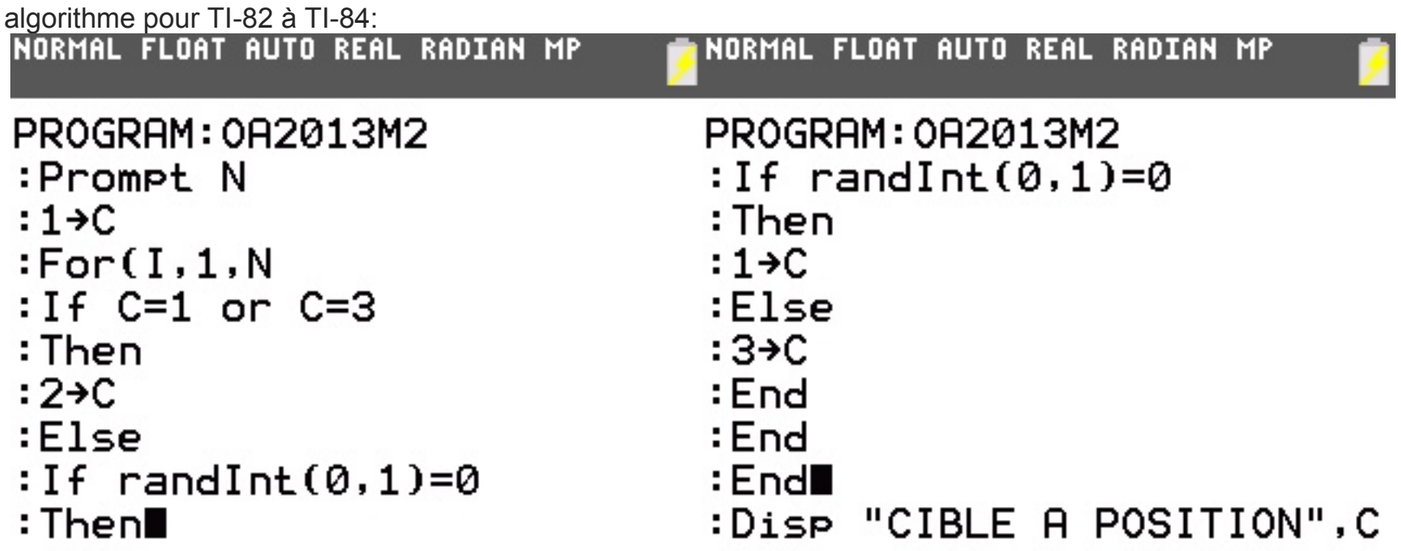
Voici l'algorithme:

Pack et mise en page par Laurae, rédigé par critor sur le forum, administrateurs de TI-Planet.org

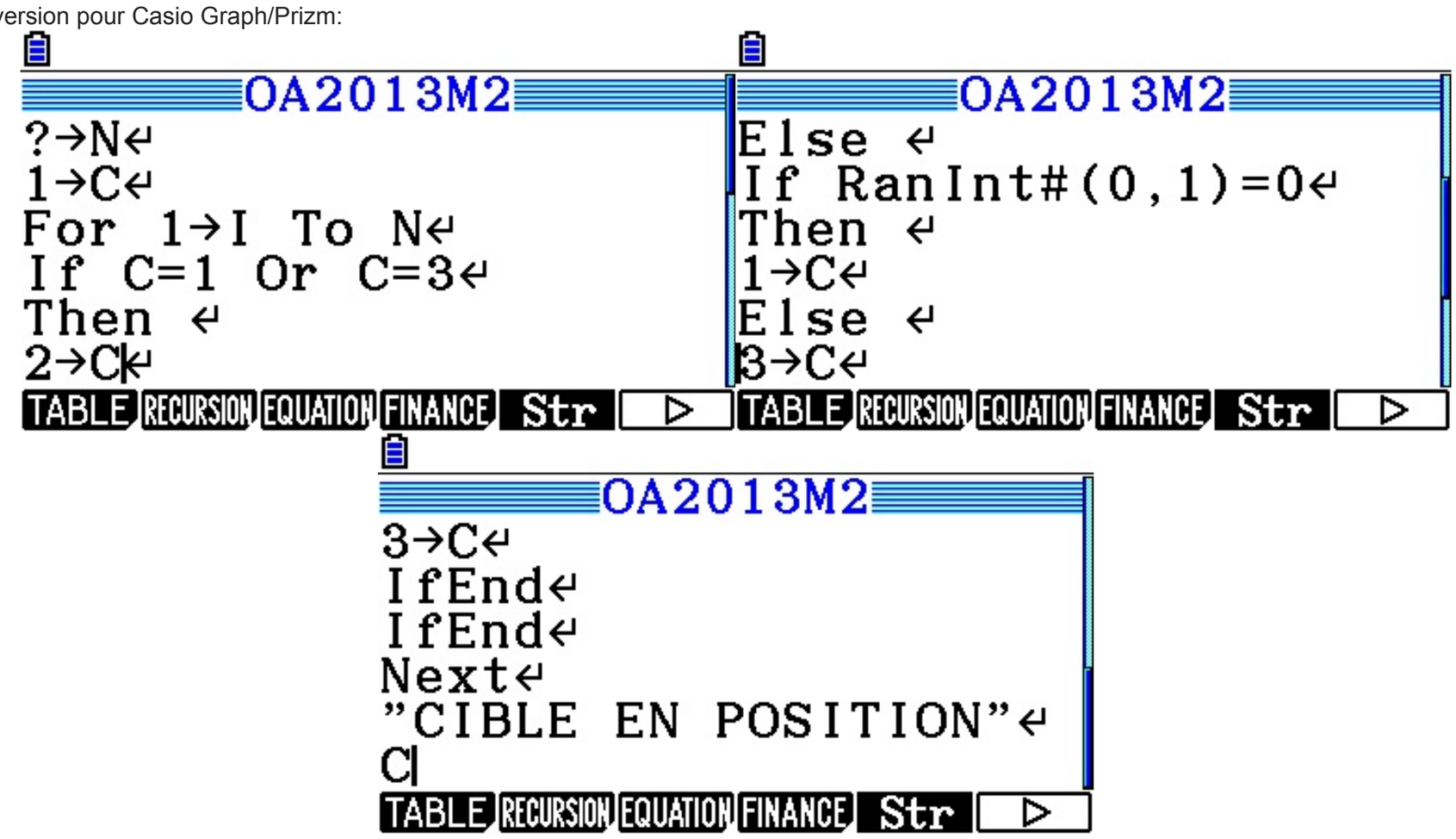
```
Variables: n, C, i entiers
Début
   Entrer n
   C←1
   Pour i=1 à n faire
      Si C=1 ou C=3 alors
         C←2
      sinon
         Si EntAlea(0,1)=0 alors
            C←1
         sinon
            C←3
         FinSi
      FinSi
   FinPour
   Afficher "la cible est en position", C
Fin
```

L'on vérifie encore une fois que l'algorithme est bon en testant sur calculatrice.

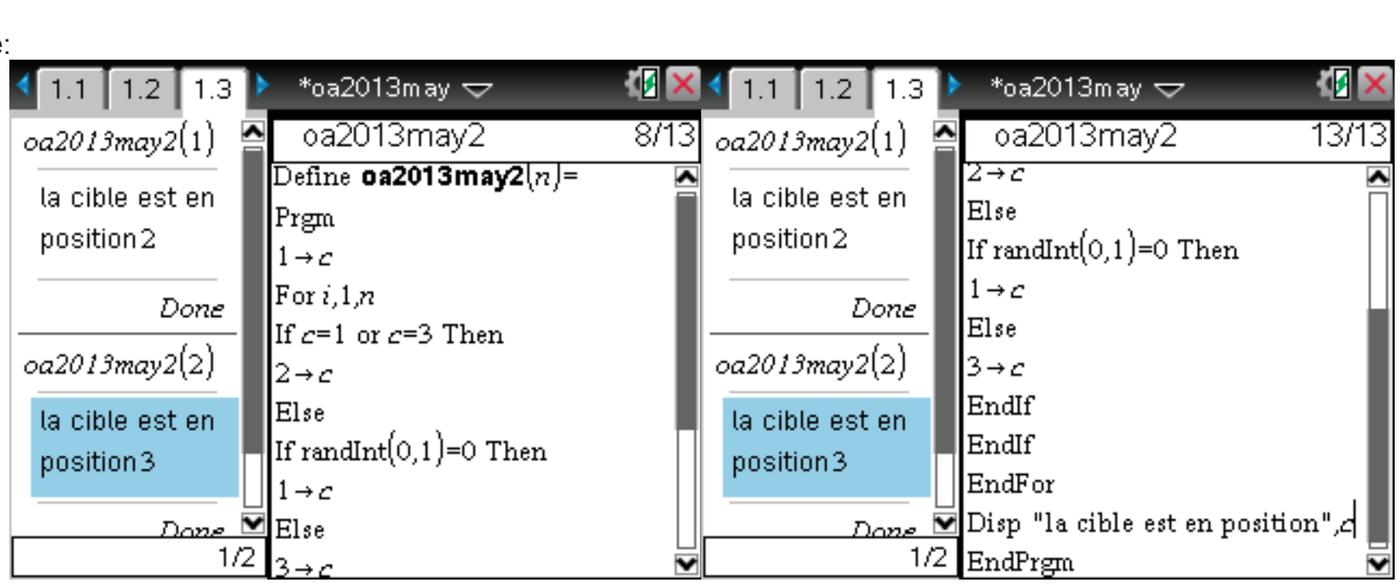
Voici un programme traduisant cet algorithme pour TI-82 à TI-84:



Voici maintenant une version pour Casio Graph/Prizm:



Et voici enfin une version TI-Nspire:



Au final, quelles sont les différences entre ces deux algorithmes?

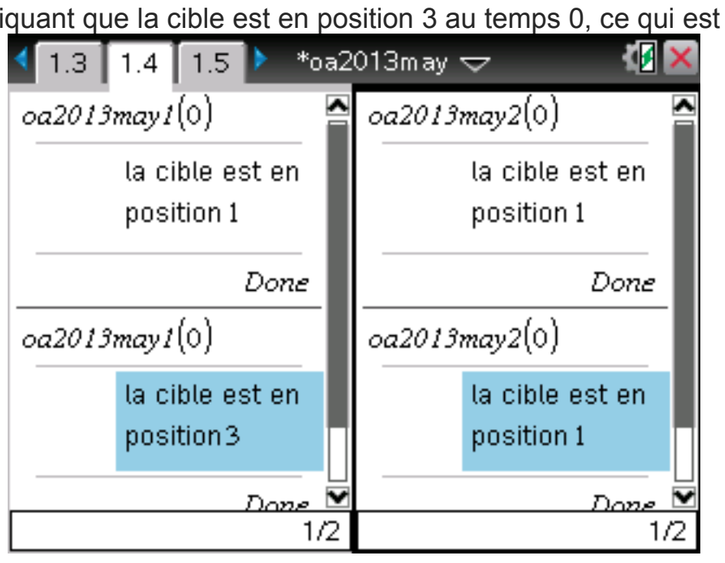
Pack et mise en page par Laurae, rédigé par critor sur le forum, administrateurs de TI-Planet.org

L'algorithme n°1 n'utilise aucune boucle. Il utilise simplement les règles de probabilité établies dans les questions précédentes.
Il s'exécute donc en un temps constant sur machine, quelle que soit la valeur de n. On dit que sa complexité est en o(1).

L'algorithme n°2 par contre a une toute autre approche et simule réellement à l'aide d'une boucle 'pour' la totalité des n déplacements de la cible.
C'est donc un algorithme linéaire de complexité en o(n), dont le temps d'exécution sur machine sera proportionnel à n.

En complexité, l'algorithme 1 serait donc meilleur et la simulation complète effectuée de l'algorithme n°2 serait inutile dans le contexte de ce qu'il doit renvoyer dans cet exercice.

Notons toutefois un petit bémol, avec le cas particulier n=0 interdit par l'énoncé.
Après 0 déplacement, la cible est par définition en position 1, la position d'origine.
L'algorithme n°2 qui simule tous les détails des déplacements est parfaitement d'accord avec ça.
Mais l'algorithme n°1 se plante une fois sur deux, en nous expliquant que la cible est en position 3 au temps 0, ce qui est impossible!



Pack et mise en page par Laurae, rédigé par critor sur le forum, administrateurs de TI-Planet.org

Correction 22 : Correction algo Olympiades Académiques 2013 1èES/L/Tech Nice

de **critor** » 11 Mai 2013 13:38

Salut à tous.

En attendant les prochains sujets de BAC fin mai, aujourd'hui continuons à réviser en nous entraînant en algorithmique et programmation avec l'algorithme qui est tombé en exercice 3 pour les Premières ES, L et Technologiques aux Olympiades Académiques dans l'Académie de Nice:

Exercice 3 : le château de cartes

Savez-vous faire un château de cartes ?

Pour arriver à un étage, c'est tout simple :

Pour deux étages, ce n'est pas très compliqué non plus :

1.

a) Combien de cartes sont nécessaires pour construire trois étages ?

b) Justifier qu'il faut utiliser exactement 26 cartes pour construire quatre étages.

2. On donne le tableau suivant :

Nombre d'étages	1	2	3	4	5
Nombre de cartes utilisées	2	7		26	40

Pour tout entier n plus grand que 1, on note $C(n)$ le nombre de cartes utilisées pour construire un château à n étage(s).

Déterminer les nombres a et b en admettant que pour tout entier n supérieur ou égal à 1 on a $C(n) = an^2 + bn$.

3. À quoi peut bien servir l'algorithme ci-dessous dans le cadre de cet exercice ?

Variables : n et c sont des entiers naturels

Traitement : Demander à l'utilisateur la valeur de n.

Affecter à c la valeur $0,5 \times n \times (3 \times n + 1)$

Sortie : Afficher c.

4.

a) Écrire un algorithme qui calcule et affiche la taille du plus grand château que l'on peut fabriquer si l'on dispose de 500 cartes.

b) Retrouver, à l'aide d'un calcul, le résultat affiché par l'algorithme.

On étudie donc le nombre de cartes nécessaires pour construire un château à n étages.

Question 3:
On nous demande donc à quoi peut bien servir l'algorithme fourni.
Cette question arrivant rapidement en début d'énoncé, on peut se douter qu'il s'agit d'un calcul du nombre de cartes nécessaires.

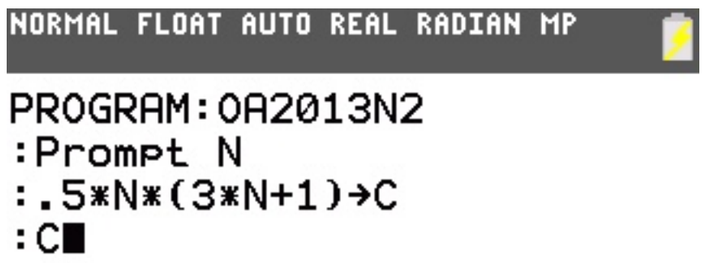
Cet algorithme utilise de plus une expression factorisée qu'il suffit de développer pour retrouver la formule normalement normalement déterminée à la question 2: $1,5n^2+0,5n$.

Et même si l'on n'arrivait pas à voir cela, il suffisait de programmer cet algorithme sur nos calculatrices graphiques, et de se rendre compte que les résultats fournis étaient en accord avec ceux du tableau de valeurs de la question 2.

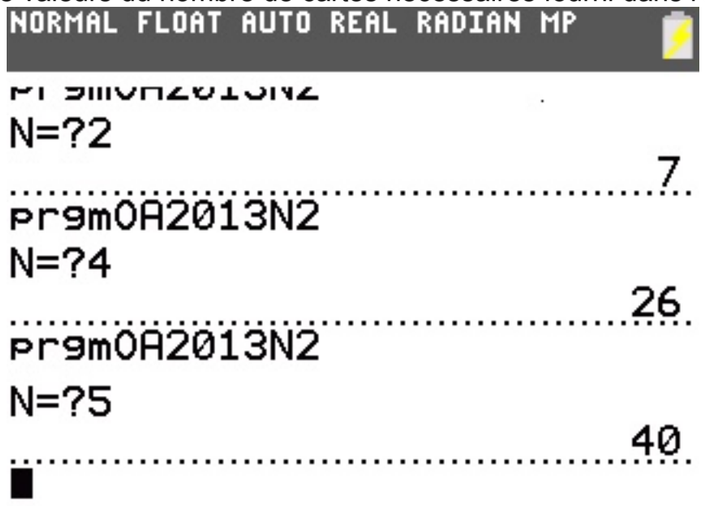
Correction algorithmes

Pack et mise en page par Laurae, rédigé par critor sur le forum, administrateurs de TI-Planet.org

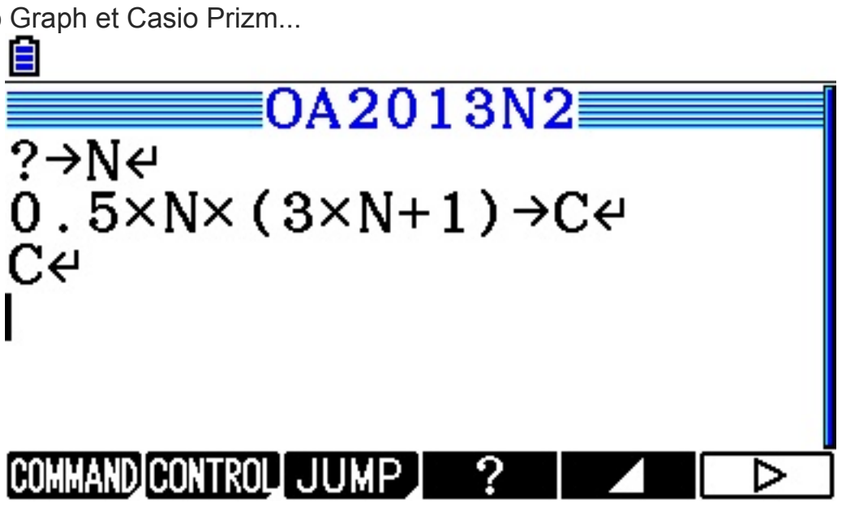
Voici le programme pour toutes les calculatrices TI-82 à TI-84:



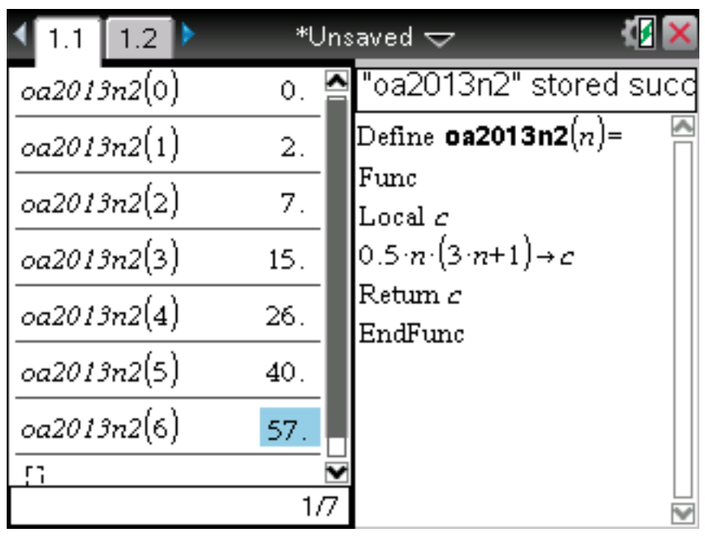
Et effectivement, les résultats sont en accord avec le tableau de valeurs du nombre de cartes nécessaires fourni dans l'énoncé:



La même chose est réalisable sur nos calculatrices Casio Graph et Casio Prizm...



Ou même encore sur nos TI-Nspire:



Question 4)a)
Il nous faut donc maintenant créer un algorithme permettant de déterminer le nombre d'étages réalisables avec 500 cartes.

Je vous propose un algorithme autour d'une boucle 'tant que', qui va compter le nombre d'étages en incrémentant un compteur n, tant que l'on ne dépasse pas 500 cartes:

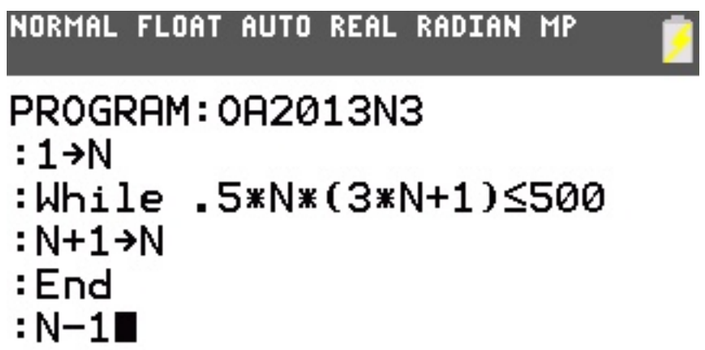
CODE: TOUT SÉLECTIONNER

```
Traitement:
   n prend la valeur 1
   Tant que 0,5n(3n+1)≤500
      n prend la valeur n+1
   Fin tant que
Sortie:
   Afficher n-1
```

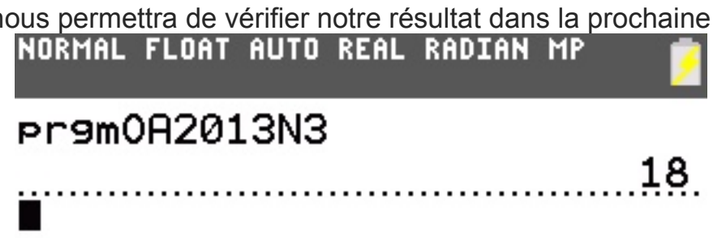
La boucle s'arrêtant lorsque le nombre de cartes 0,5n(3n+1) dépasse strictement 500, en fin de boucle le résultat à afficher n'est pas n mais n-1.

Voici un programme implémentant cet algorithme pour toutes TI-82 à TI-84:

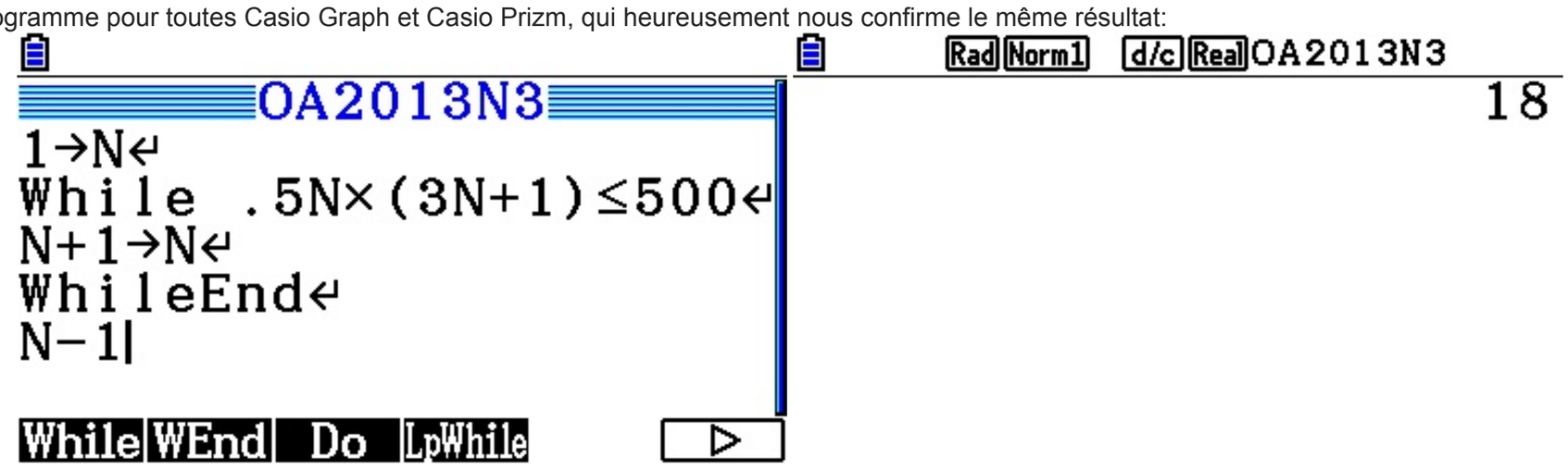
Pack et mise en page par Laurae, rédigé par critor sur le forum, administrateurs de TI-Planet.org



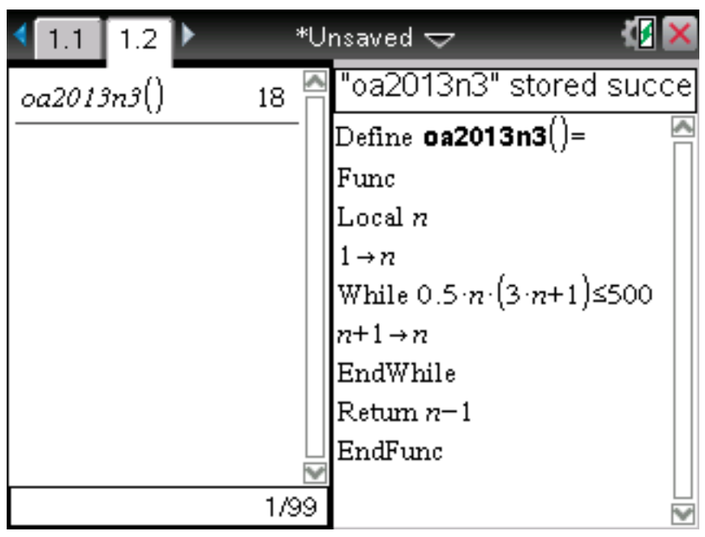
Le programme nous donne même la réponse, 18 étages, qui nous permettra de vérifier notre résultat dans la prochaine question 4)b):



Voici maintenant le programme pour toutes Casio Graph et Casio Prizm, qui heureusement nous confirme le même résultat:



Et voici enfin le programme pour toutes TI-Nspire:



A bientôt!

Correction 23 : Correction algo Olympiades Académiques 2013 1°S Montpellier

de **critor** » 15 Mai 2013 23:21

Ce soir, continuons de nous entraîner pour nos examens en regardant l'algorithme qui est tombé en exercice 3 aux Olympiades Académiques 2013 pour les Premières S de l'Académie de Montpellier:

Sur la figure ci-dessus, ABCD est un rectangle et BEC est un triangle rectangle. On donne les longueurs AB = 2 et CE = 1 et on pose BE = x.

1. a. Exprimez l'aire $f(x)$ du trapèze en fonction de x.

   b. Tracez la courbe de f à l'aide de votre calculatrice. Donnez, à 10^{-1} près, la valeur du maximum M de f ainsi que l'abscisse x_M correspondante.

2. Un algorithme.
Dans le tableau ci-dessous qui décrit un algorithme, le symbole * représente la multiplication ; **racine** représente la racine carrée ; l'écriture >= représente le symbole de l'inégalité ≥.

Variables	x, y, p sont des réels ; n est un entier.
Données	x prend la valeur 0, y prend la valeur −1, p prend la valeur 0,01. n prend la valeur 0.
Algorithme	Tant que (0,5*racine((1 − x^2)*(x + 4)^2) >= y) faire y prend la valeur 0,5*racine((1 − x^2)*(x + 4)^2) x prend la valeur x + p n prend la valeur n + 1 fin Tant que
Sortie	Afficher x, y, n

a. Montrez que l'algorithme se termine.
b. Donnez les valeurs affichées en sortie.

Il y a ici une petite originalité, car en plus de l'algorithme il y a également une conjecture à réaliser sur calculatrice.

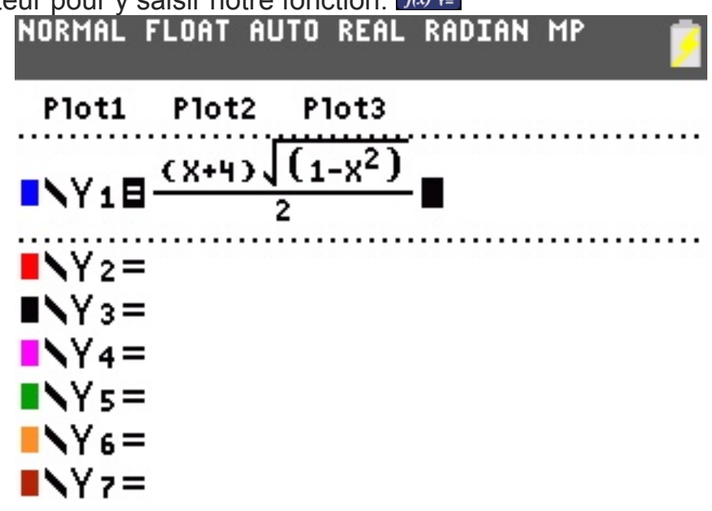
Question 1)a)
Nous devons donc exprimer l'aire du trapèze AECD.
La formule de l'aire du trapèze donne ici (AE+CD)*AD/2.
Mais en cas de trou de mémoire, on peut toujours répondre à la question en additionnant les aires des rectangle ABCD et triangle rectangle BCE.

CD=AB=2
AE=AB+BE=4+x
AD=BC=$\sqrt{(CE^2-BE^2)}$=$\sqrt{(1-x^2)}$

Nous obtenons donc f(x)=(x+4)$\sqrt{(1-x^2)}$/2.

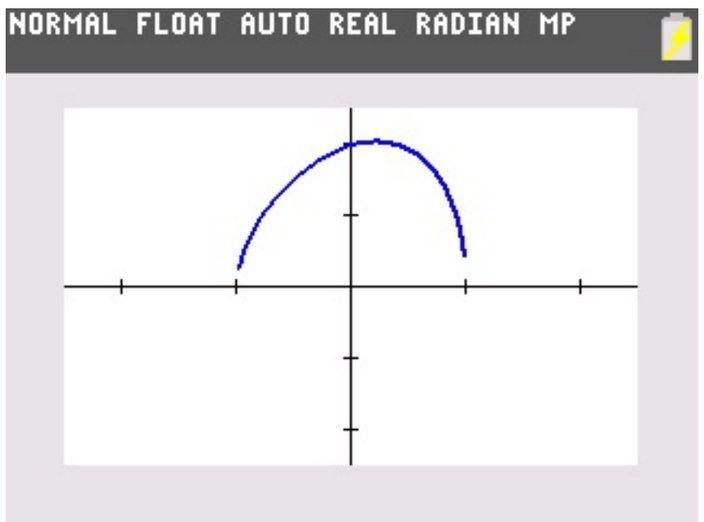
Question 1)b)
On nous demande maintenant de conjecturer la valeur du maximum de f à l'aide de la calculatrice.

Prenons donc notre calculatrice TI-82 à TI-84, et ouvrons l'éditeur pour y saisir notre fonction:

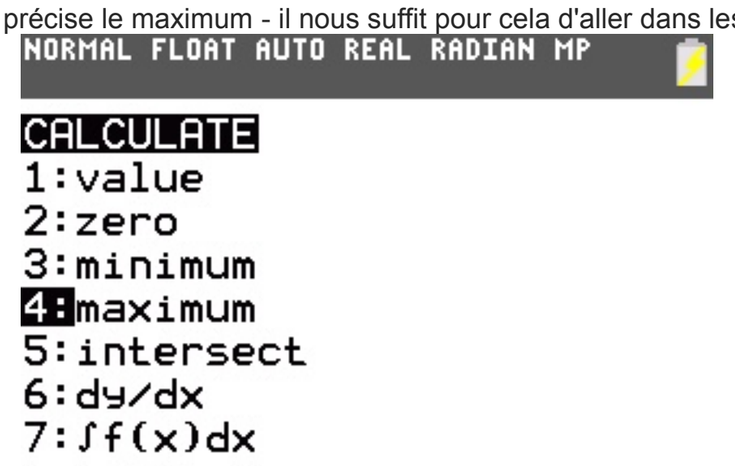


Pack et mise en page par Laurae, rédigé par critor sur le forum, administrateurs de TI-Planet.org

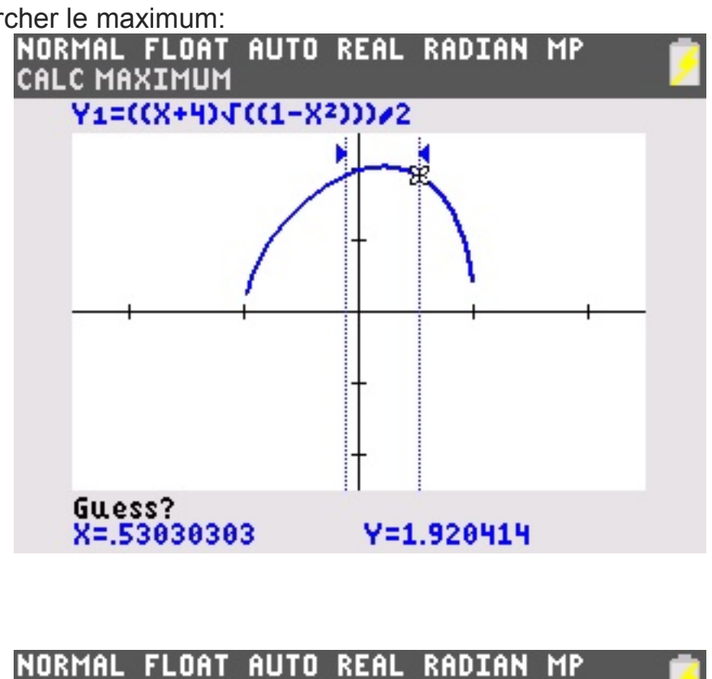
Nous pouvons alors demander sa courbe: table graphe F5 GRAPH



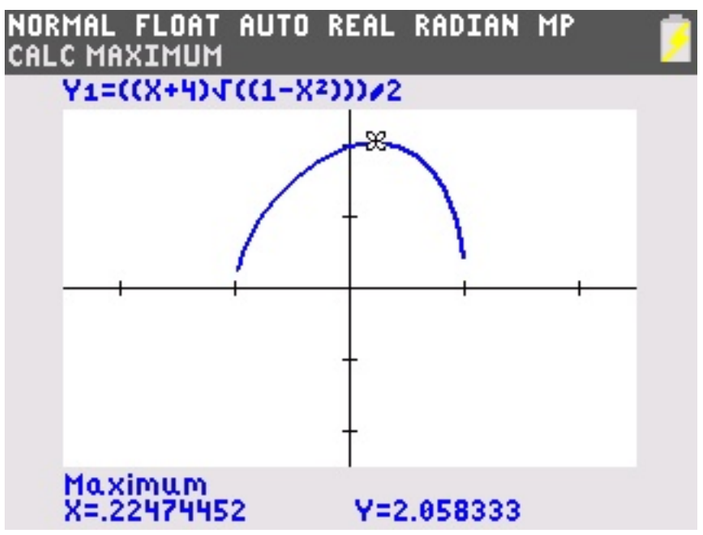
Notre calculatrice dispose d'un outil pour rechercher de façon précise le maximum - il nous suffit pour cela d'aller dans les outils d'analyse de courbe: 2nde 2ND calculs CALC F4 trace



Nous spécifions alors graphiquement sur quel intervalle rechercher le maximum:



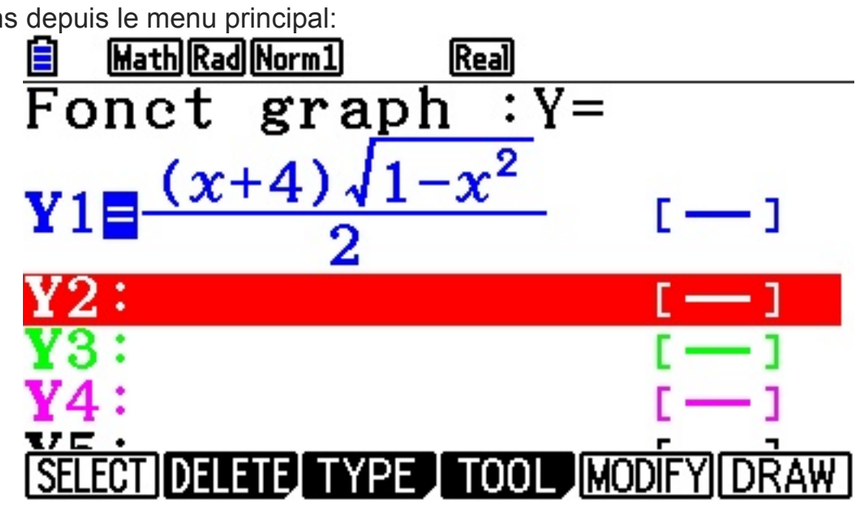
Et le voici enfin, maximum d'environ 2,1 pour x≈0,2:



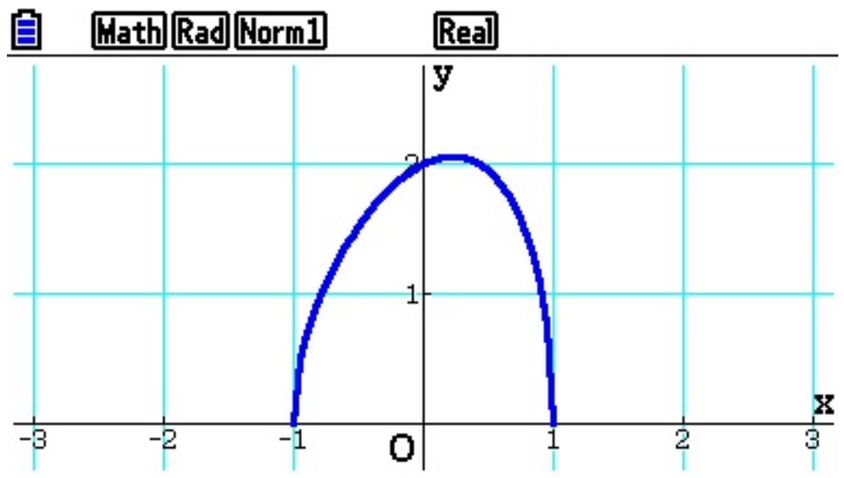
Pack et mise en page par Laurae, rédigé par critor sur le forum, administrateurs de TI-Planet.org

Si vous disposez d'une Casio Graph ou Casio Prizm, on procède selon les mêmes étapes.

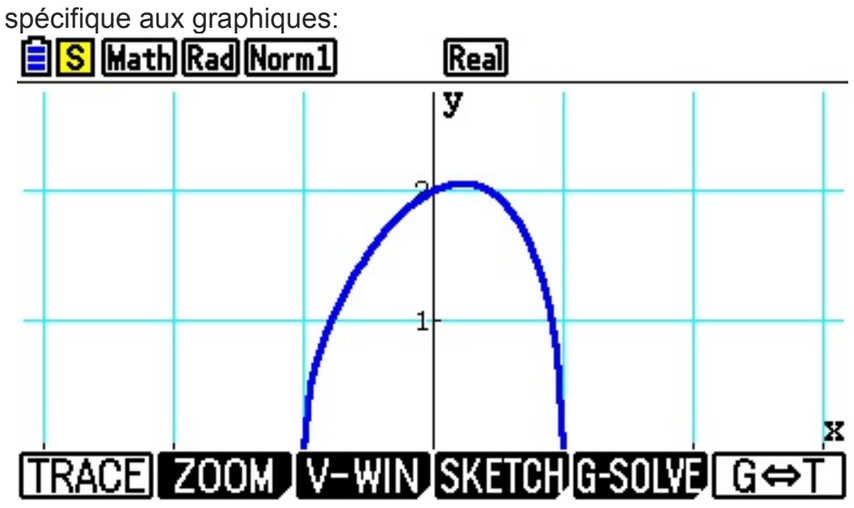
On saisit l'expression en accédant à l'éditeur de fonctions depuis le menu principal:



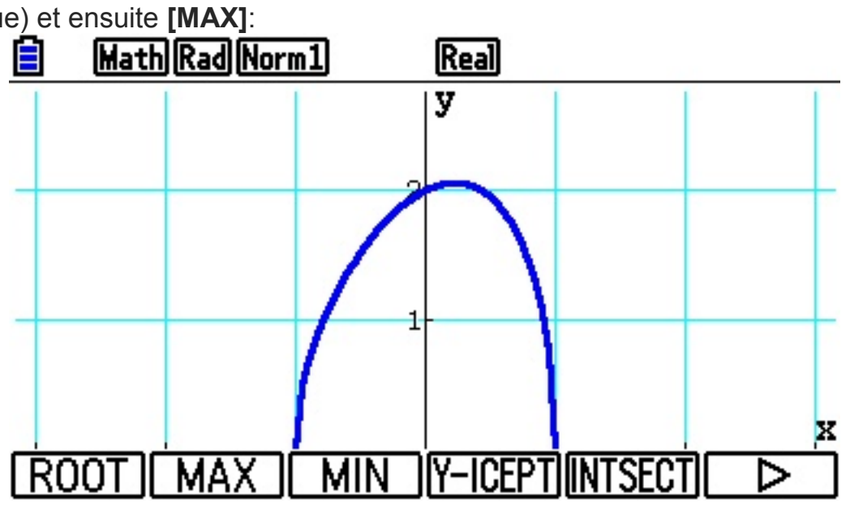
Avec **[DRAW]**, on en demande la courbe:



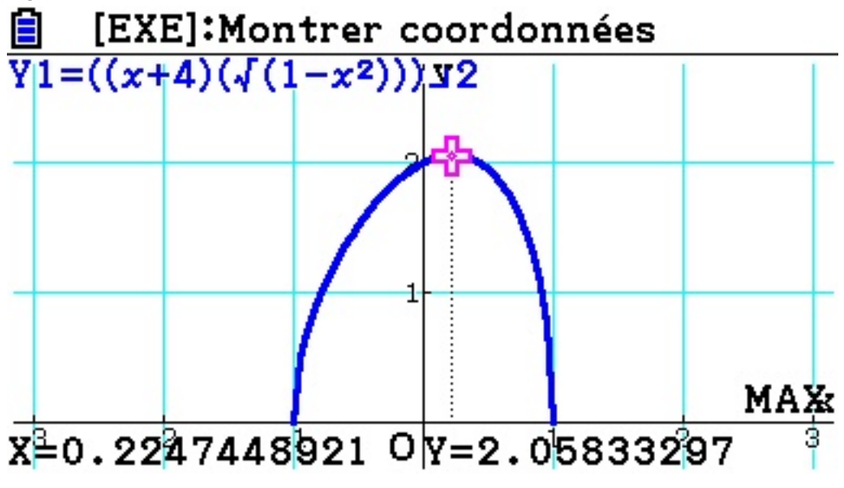
En appuyant sur **[SHIFT]**, nous obtenons alors un menu spécifique aux graphiques:



Nous y choisissons **[G-SOLVE]** (pour résolution graphique) et ensuite **[MAX]**:

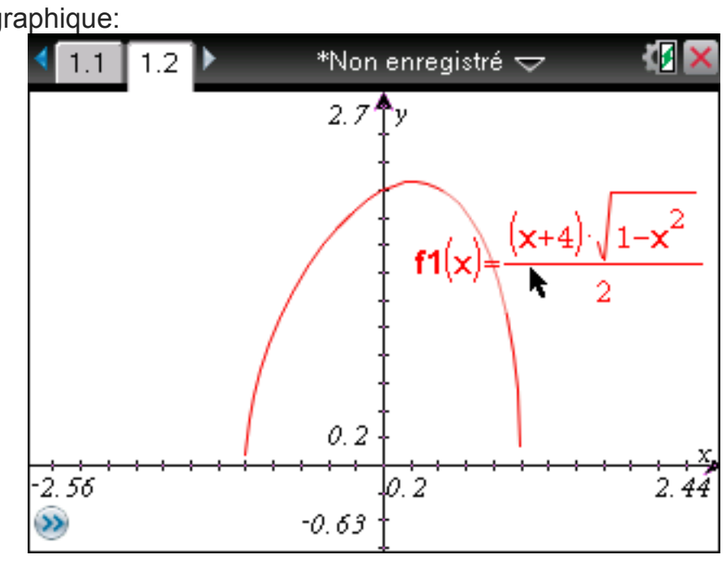


Et voilà, même résultat - il n'y a pas d'intervalle à spécifier ici:

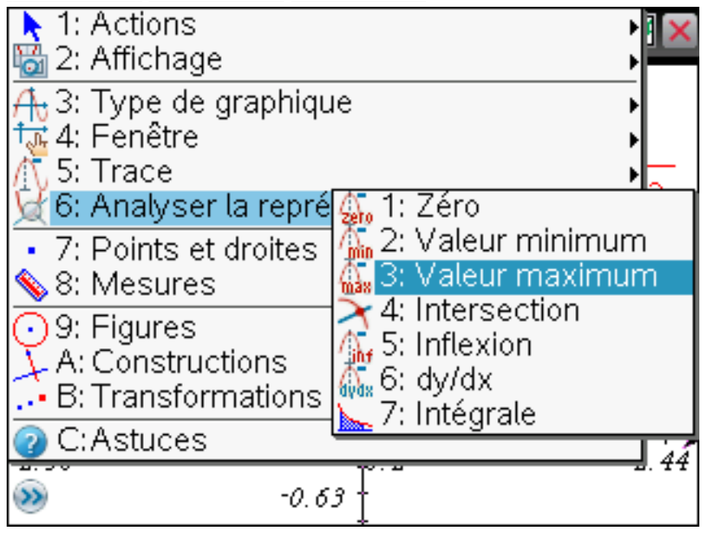


Pack et mise en page par Laurae, rédigé par critor sur le forum, administrateurs de TI-Planet.org

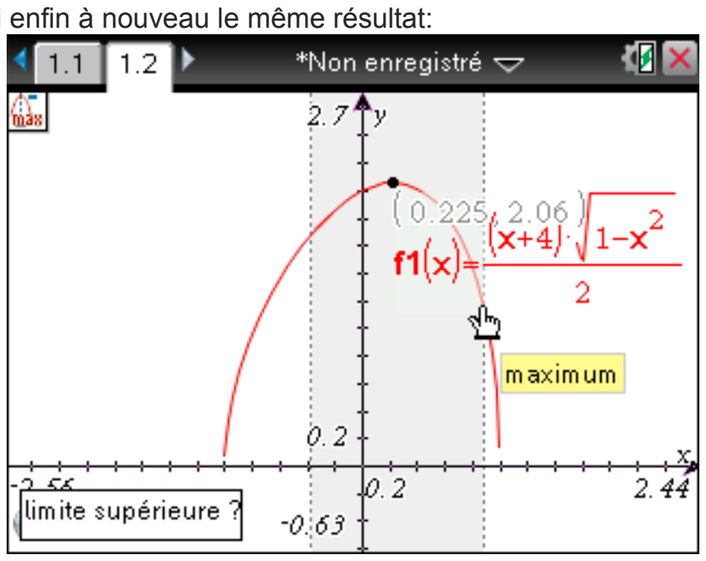
Sur TI-Nspire, nous saisirons l'expression dans l'application graphique:



Nous choisissons ensuite l'outil d'analyse de courbe approprié: menu 6 3



Après spécification graphique de l'intervalle de recherche, voici enfin à nouveau le même résultat:



Question 2)a)

On nous donne donc un algorithme qui travaille sur l'expression $0,5\sqrt{((1-x^2)(x+4)^2)}$.
Mais remarquons que $0,5\sqrt{((1-x^2)(x+4)^2)}=0,5\sqrt{((x+4)^2)}\sqrt{(1-x^2)}=0,5(x+4)\sqrt{(1-x^2)}$ car $x+4\geq$ puisque $x\geq 0$ (x étant la longueur BE).
Cette expression revient dont finalement à f(x) et représente l'aire du trapèze.

Que fait donc l'algorithme là-dessus?
Il balaye les valeurs en partant de 0 par pas de 0,01 et continue tant que l'image est supérieure ou égale à la précédente.

En gros il recherche la plus grande image, soit le maximum.

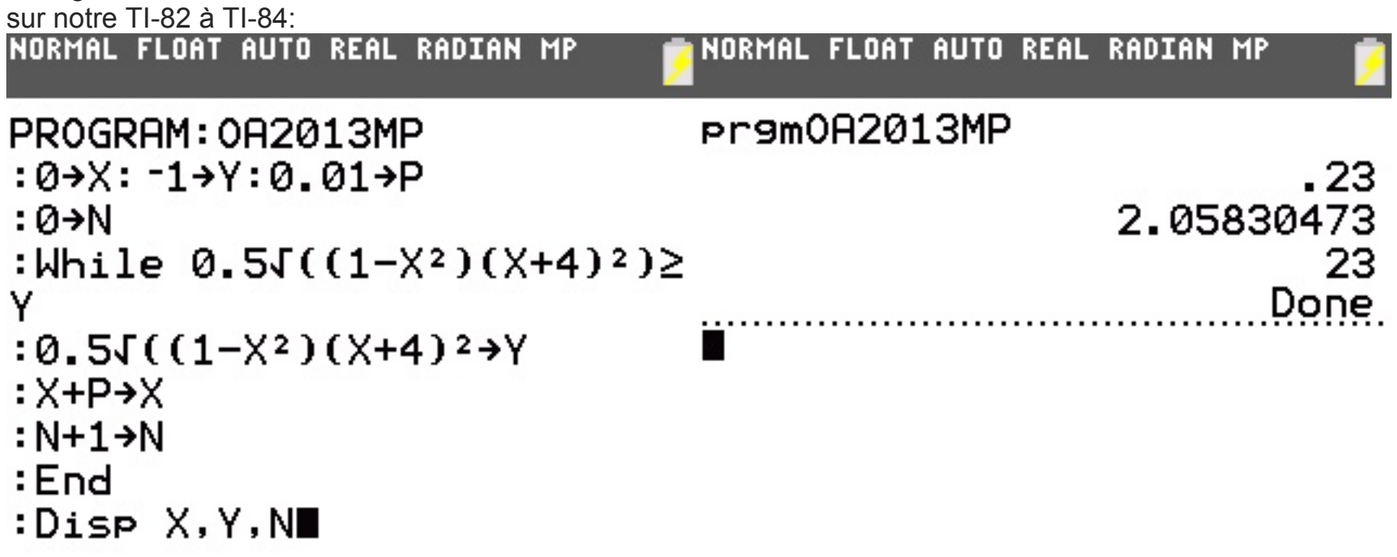
De façon plus précise, comme on utilise un pas de 0,01 on peut dire que l'algorithme recherche l'abscisse du maximum à 10^{-2} près.

Mais en fait, l'algorithme s'arrêtera uniquement lorsque l'on aura dépassé ce maximum en abscisse.
Contrairement à la question précédente, on pourrait donc dire que l'algorithme recherche ici à 10^{-2} près par excès, la valeur de x correspondant au maximum.

Le maximum existant ici, la recherche se termine.

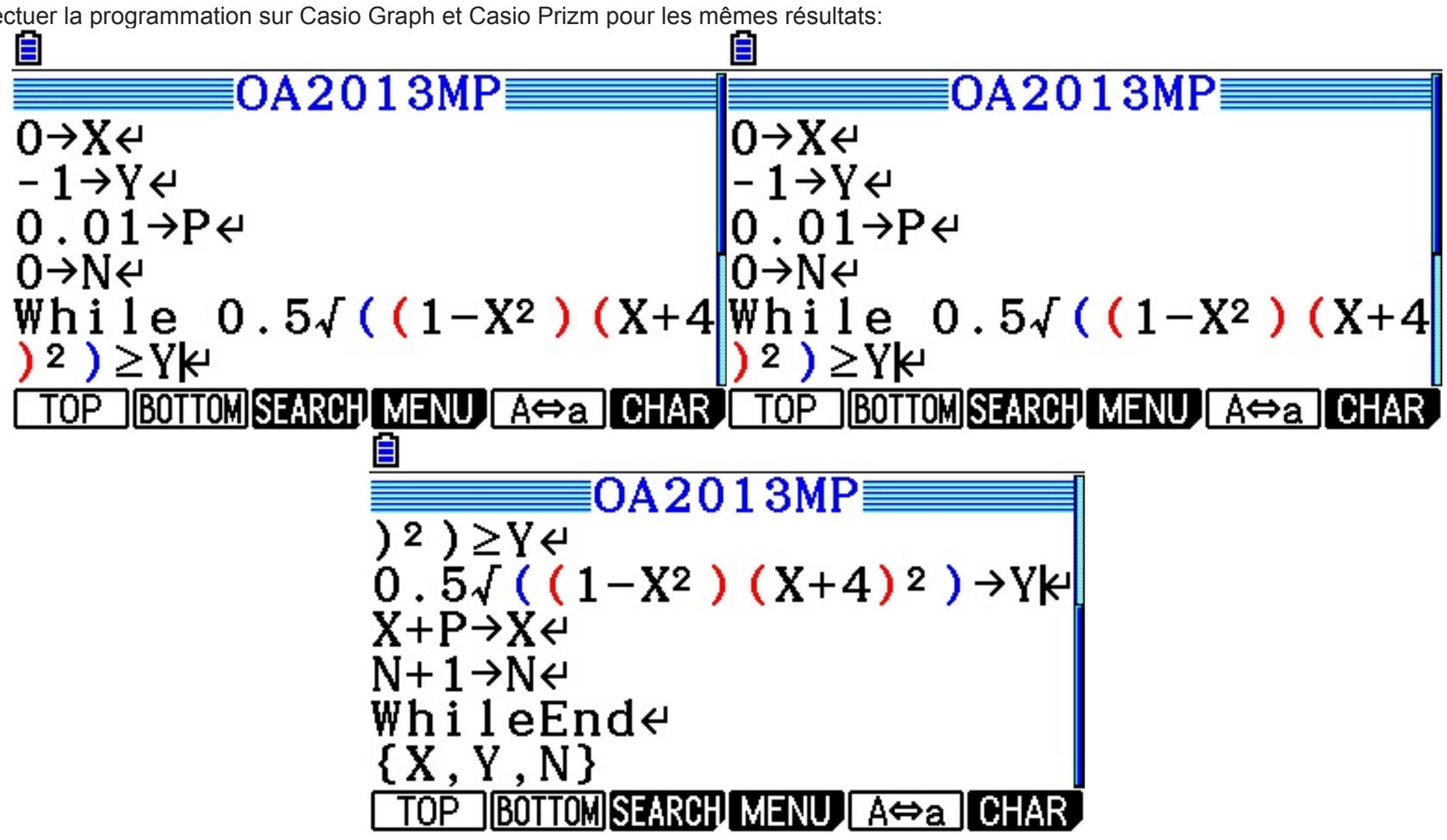
Question 2)b)

On nous demande enfin ce qu'affiche l'algorithme.
Il suffit pour cela de le programmer sur notre TI-82 à TI-84:

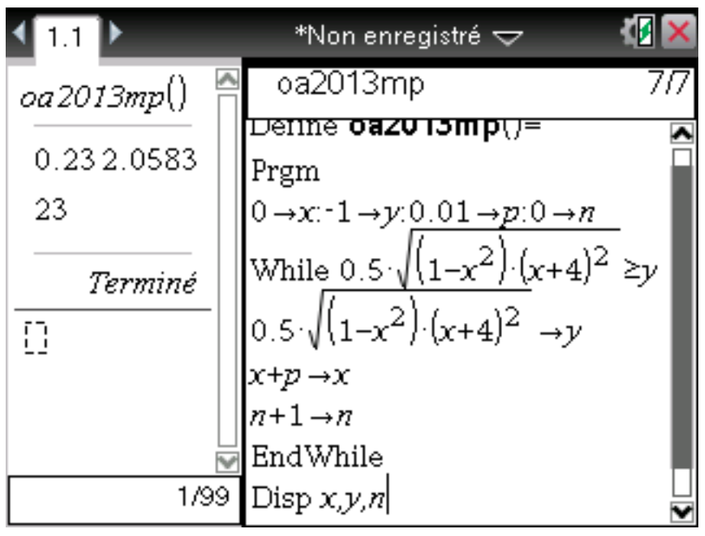


On obtient donc dans l'ordre 0,23, 2,058300473 et 23.

On peut également effectuer la programmation sur Casio Graph et Casio Prizm pour les mêmes résultats:



Ou encore sur TI-Nspire:



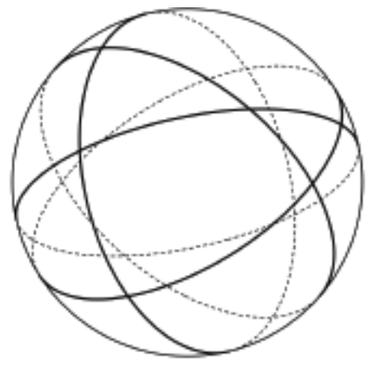
Pack et mise en page par Laurae, rédigé par critor sur le forum, administrateurs de TI-Planet.org

Correction 24 : Correction algo Olympiades Académiques 1èreS Toulouse

de **critor** » 25 Mai 2013 14:34

Voici aujourd'hui l'algorithme tombé en exercice 6 aux Olympiades Académiques de 1èreS dans l'Académie de Toulouse.

On considère quatre points M_1, M_2, M_3 et M_4 choisis aléatoirement sur la Terre.
On note H_{M_1}, H_{M_2}, H_{M_3}, H_{M_4} les hémisphères de pôles M_1, M_2, M_3, M_4.
On suppose que les grands cercles associés aux pôles M_1, M_2, M_3, M_4 sont deux à deux sécants. On exclut tout autre type d'intersection de ces quatre cercles, on admet qu'il s'agit de situations ne changeant pas le calcul de probabilité.
Ces grands cercles découpent la surface de la Terre en diverses « régions ».



1) On choisit au hasard une des « régions » ainsi définies. Quelle est la probabilité qu'elle soit entièrement située dans l'intersection des quatre hémisphères H_{M_1}, H_{M_2}, H_{M_3}, H_{M_4} ?

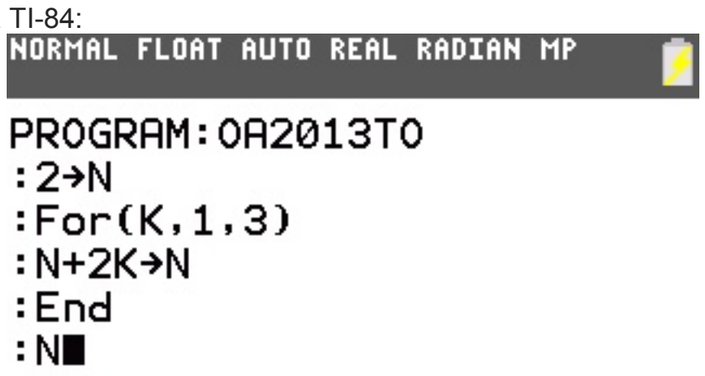
2) L'algorithme ci-contre permet de dénombrer le nombre de régions obtenues.
a) A l'aide de cet algorithme, déterminer le nombre de régions.
b) Expliquer pourquoi l'algorithme précédent dénombre effectivement le nombre de régions.

Variable : **NbRegion**
Initialisation :
On affecte 2 à **NbRegion**
Traitement :
Pour *k* allant de 1 à 3
On affecte **NbRegion** + 2*k* à **NbRegion**
Fin

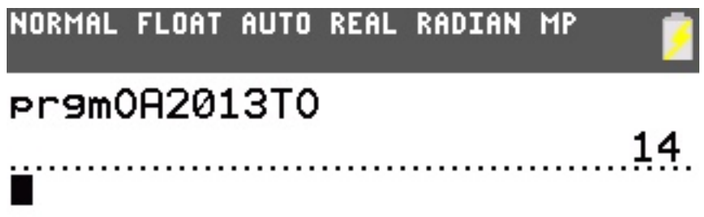
Il s'agit donc d'un partage d'une sphère selon de grands cercles (sections passant par son centre).
On suppose dans cet énoncé, que les grands cercles sont tous sécants deux à deux.

Question C)2)a):
On nous propose donc un algorithme permettant de déterminer le nombre de régions obtenues après quatre sections de la sphère.
Il nous suffit donc de programmer cet algorithme sur notre calculatrice graphique afin d'obtenir la réponse.

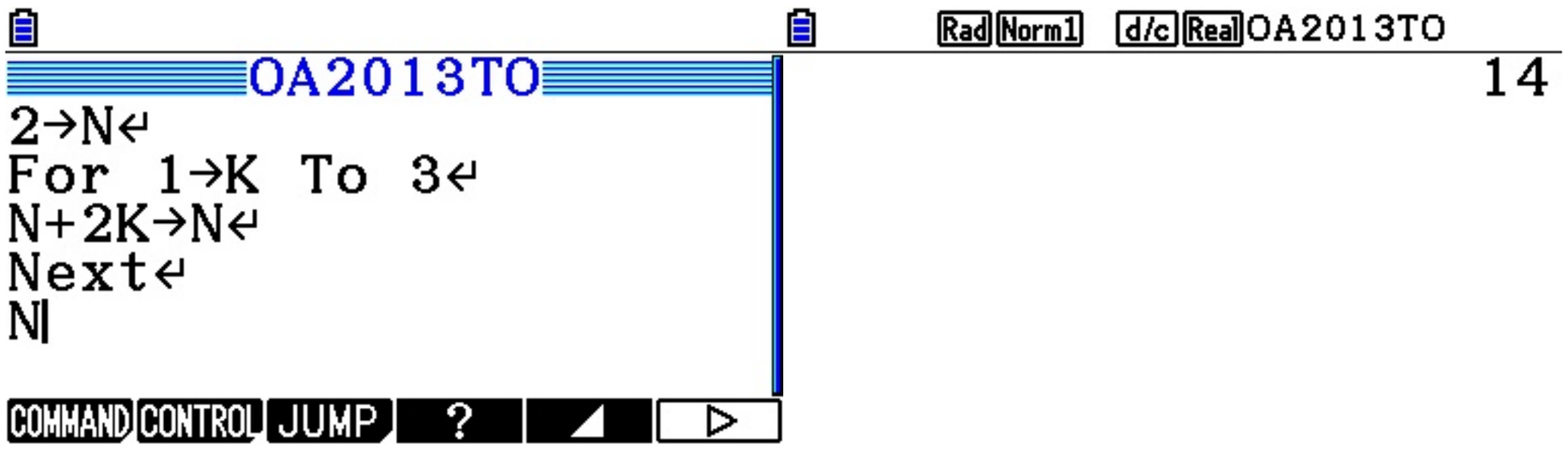
Voici sa traduction en un programme pour calculatrices TI-82 à TI-84:



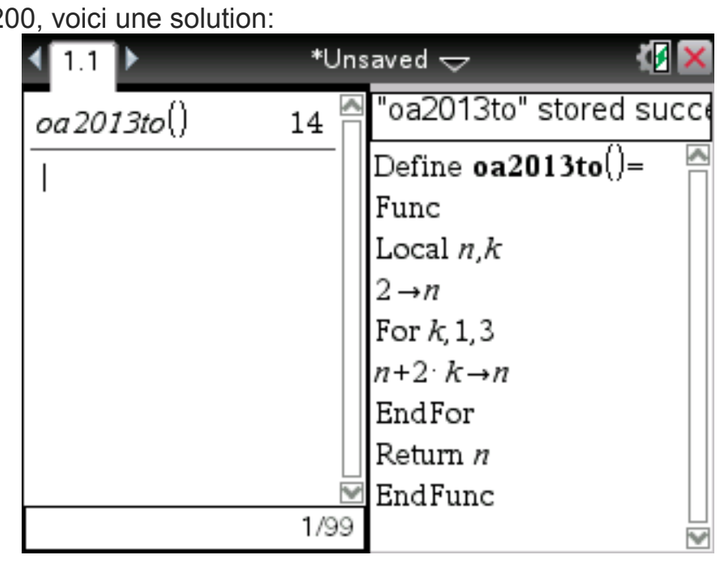
L'exécution du programme nous donne alors la réponse, 14:



Si vous êtes équipés d'une calculatrice Casio Graph/Prizm, vous pouvez sans difficulté réaliser exactement la même chose:



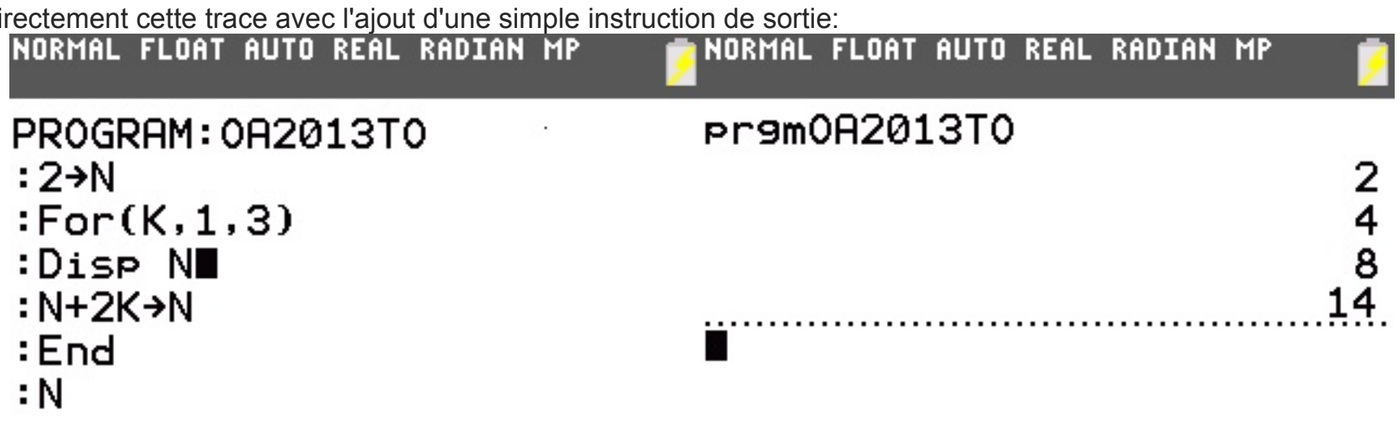
Si vous êtes dotés d'une calculatrice TI-Nspire ou TI-89/92/V200, voici une solution:



Question C)2)b):
L'on nous demande maintenant d'expliquer pourquoi cet algorithme répond bien à la question.

Pour mieux comprendre ce qu'il fait, on pourrait construire la trace de l'algorithme, avec toutes les valeurs intermédiaires prises par la variable NbRegion à chaque passage dans la boucle.

La calculatrice peut nous donner directement cette trace avec l'ajout d'une simple instruction de sortie:



De l'initialisation à la sortie du programme, la variable N prend donc successivement les valeurs 2, 4, 8, 14.

C'est-à-dire que l'on ajoute successivement 2, 4 et 6.

En effet, la valeur initiale 2 est le nombre de régions déterminées par une section selon un grand cercle.

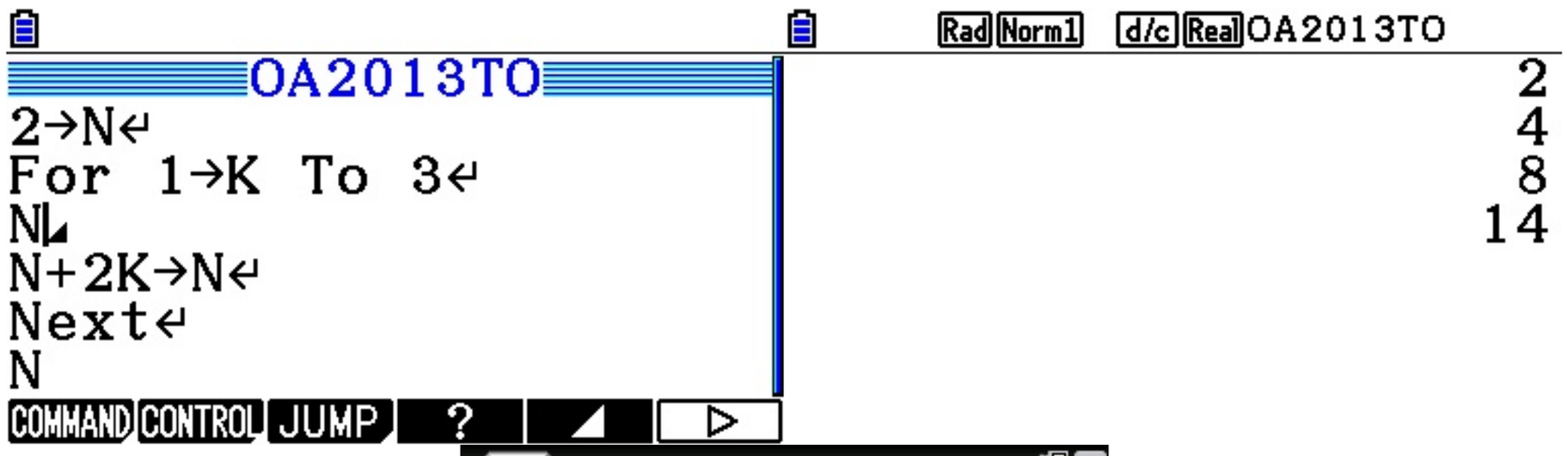
A la 2ème section, les grands cercles étant sécants deux à deux, les deux régions précédentes sont coupées en deux, ce qui nous rajoute 2 régions.

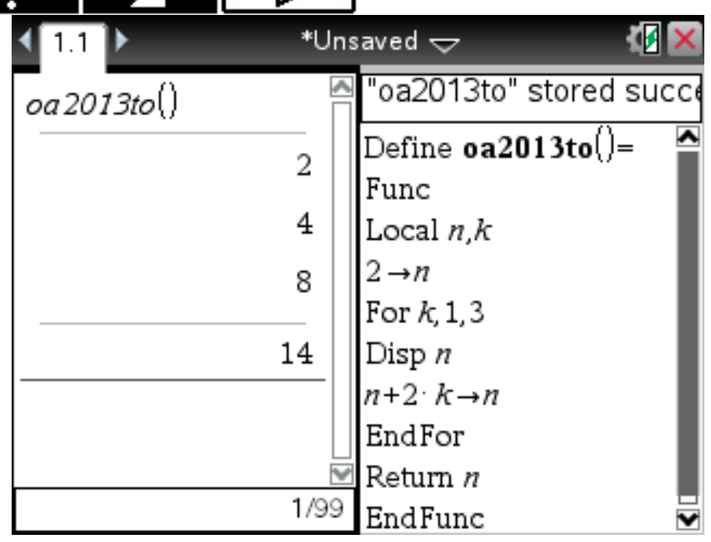
A la 3ème section, le nouveau grand cercle coupe les deux grands cercles précédents, ce qui rajoute 4 régions.

A la 4ème section, le nouveau grand cercle couple les trois grands grands cercles précédents, ce qui rajoute 6 régions.

Le même raisonnement peut être initié à partir d'une Casio Graph /Prizm ou d'une TI-Nspire/89/92/V200:

Pack et mise en page par Laurae, rédigé par critor sur le forum, administrateurs de TI-Planet.org





Correction 25 : Correction algo Olympiades Académiques 2013 1èreS Nice

de **critor** » 25 Mai 2013 16:42

Intéressons-nous maintenant à l'algorithme qui est tombé en exercice 3 aux Olympiades Académiques 2013 de Première S dans l'Académie de Nice.

5. On code un lotissement à l'aide d'un tableau `T[i,j]` où i (resp. j) désigne le numéro de la ligne (resp. colonne) avec $1 \leq i \leq 5$ et $1 \leq j \leq 5$. On convient que :

   - `T[i,j]` vaut 0 si la case $(i\,;j)$ n'est pas éclairée ;
   - `T[i,j]` vaut 1 si la case $(i\,;j)$ est éclairée ;
   - `T[i,j]` vaut 2 si la case $(i\,;j)$ contient une maison.

   On suppose un lotissement entièrement codé par un tableau T.

   Écrire un algorithme qui affiche si oui ou non le lotissement est entièrement éclairé.

6.

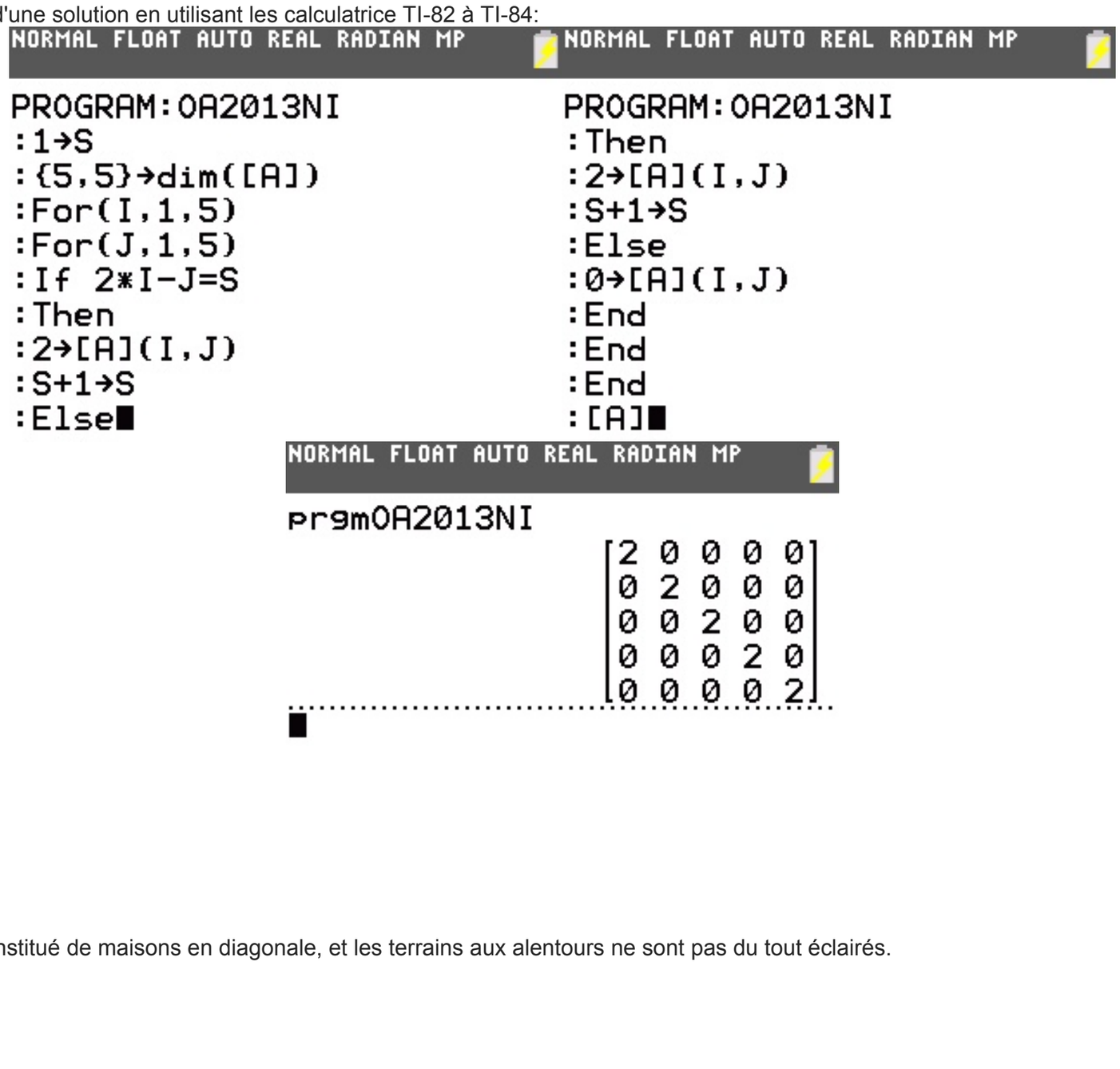
a) Dessiner le plan du lotissement associé à un tableau T obtenu à l'aide de l'algorithme ci-dessous :

```
Variables :                 i, j et s sont des entiers naturels
                            T est un tableau
Initialisation :            Affecter à s la valeur 1.
Traitement :                Pour i variant de 1 à 5
                                Pour j variant de 1 à 5
                                    Si (2*i-j)==s alors
                                        Affecter à T[i,j] la valeur 2
                                        s prend la valeur s+1
                                    Sinon affecter à T[i,j] la valeur 0
```

Il s'agit donc ici d'un problème d'éclairage.
Notons que là encore l'auteur du sujet fait plus de la traduction de langage de programmation que de l'algorithmie, notamment avec le test '==' pour l'égalité.

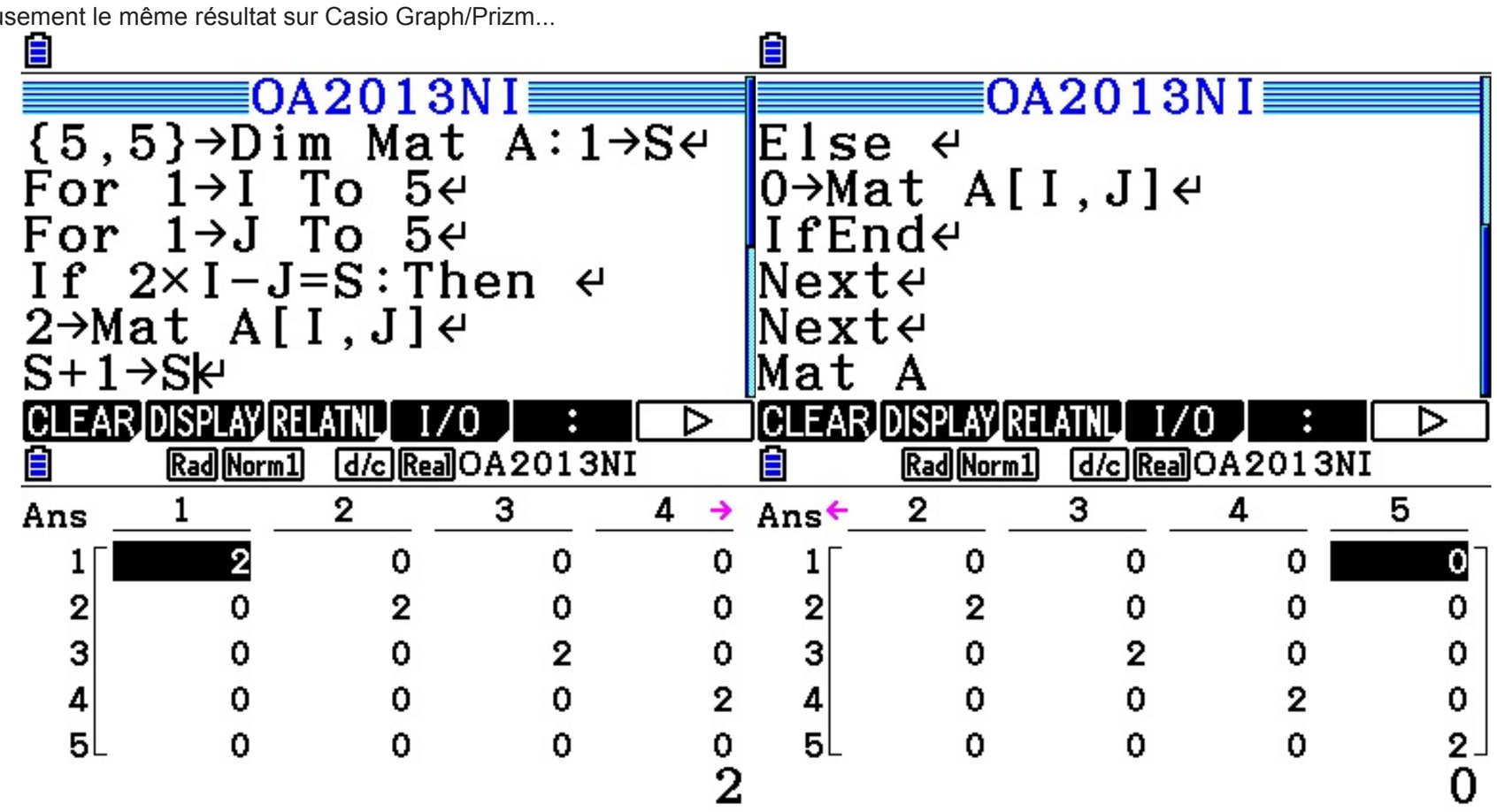
Question 6)a)
Commençons par la question 6.
L'algorithme donné nous construit un lotissement quadrillé 5x5, en codant 0 les zones d'ombre, 1 les zones éclairées, et 2 les maisons.
On nous demande un plan, et il nous suffit d'utiliser l'objet matrice (tableau de nombres vu en spécialité ES) de la calculatrice afin d'obtenir directement un schéma du lotissement.

Voici par exemple la construction d'une solution en utilisant les calculatrice TI-82 à TI-84:

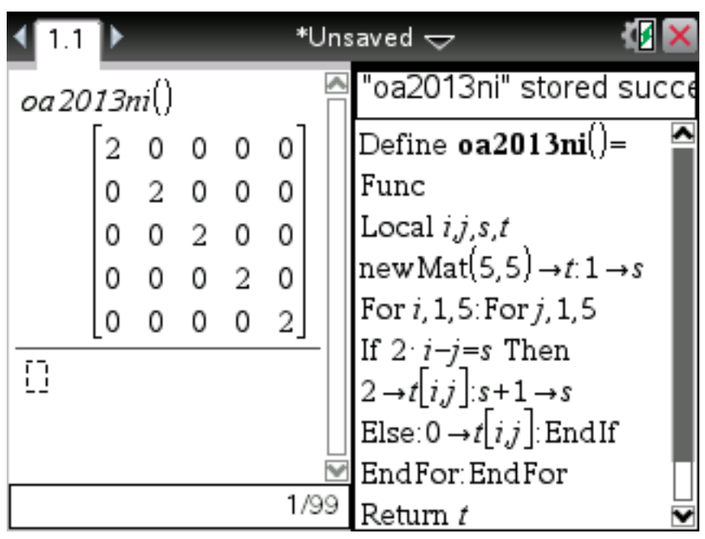


Notre lotissement 5x5 est donc constitué de maisons en diagonale, et les terrains aux alentours ne sont pas du tout éclairés.

Nous obtenons heureusement le même résultat sur Casio Graph/Prizm...



... ou encore sur TI-Nspire:



Question 5)
Maintenant que nous avons vu la question 6, nous savons mieux comment manipuler et parcourir des matrices.
Nous pouvons donc répondre à la question 5, et produire un algorithme disant si le lotissement est bien éclairé ou pas.

Deux boucles 'pour' imbriquées sur les 5 lignes et les 5 colonnes permettent de parcourir toutes les cases du tableau.

Un lotissement est mal éclairé si l'on trouve une case non éclairée (valeur 0).
Comme il peut y avoir plusieurs cases non éclairées, afin d'éviter les affichages multiples il va nous falloir stocker le résultat à l'aide d'un drapeau et l'afficher en fin de boucle.

CODE: TOUT SÉLECTIONNER

```
e prend la valeur 1
Pour i de 1 à 5
   Pour j de 1 à 5
      Si T[i,j]=0 alors
         e prend la valeur 0
      FinSi
   FinPour
FinPour
Si e=0 alors
   Afficher "Le lotissement est mal éclairé"
Sinon
   Afficher "Le lotissement est bien éclairé"
FinSi
```

Codons et testons cela tout-de-suite sur TI-82 à TI-84, avec le lotissement de la question 6:

```
PROGRAM:OA2013N2
:1→E
:For(I,1,5)
:For(J,1,5)
:If [A](I,J)=0
:Then
:0→E
:End
:End
:End
```

```
PROGRAM:OA2013N2
:End
:If E=0
:Then
:Disp "LOTISSEMENT MAL ECL
AIRE
:Else
:Disp "LOTISSEMENT BIEN EC
LAIRE
:End
```

```
prgmOA2013N2
[2 0 0 0 0]
[0 2 0 0 0]
[0 0 2 0 0]
[0 0 0 2 0]
[0 0 0 0 2]
prgmOA2013N2
LOTISSEMENT MAL ECLAIRE
Done
```

Sans surprise, ce lotissement est mal éclairé.

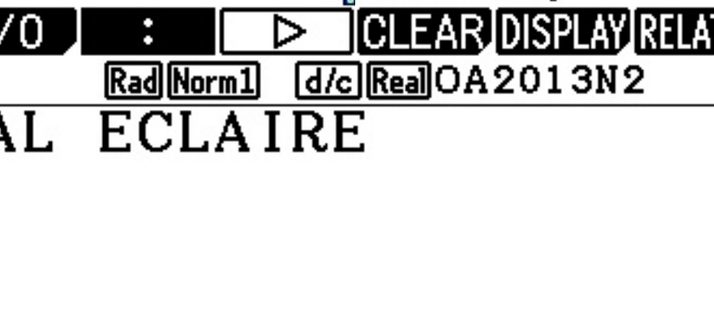
Voici maintenant le code pour Casio Graph/Prizm...

```
OA2013N2
1→E↵
For 1→I To 5↵
For 1→J To 5↵
If Mat A[I,J]=0↵
Then :0→E:IfEnd↵
Next↵
```

```
OA2013N2
Next↵
If E=0:Then ↵
"MAL ECLAIRE"↵
Else ↵
"BIEN ECLAIRE"↵
IfEnd
```

```
CLEAR DISPLAY RELATNL I/O : ▷ CLEAR DISPLAY RELATNL I/O : ▷
Rad Norm1 d/c Real OA2013N2
MAL ECLAIRE
```

... et pour TI-Nspire/89/92/V200:



Pack et mise en page par Laurae, rédigé par critor sur le forum, administrateurs de TI-Planet.org

Correction 26 : Correction algorithme Maths Obligatoire BAC S 2013 (Liban)

de **critor** » 28 Mai 2013 20:14

Ce matin, les candidats au BAC S des lycées français du Liban ont sans surprise eu droit à un algorithme en exercice 4 de leur sujet de Maths Obligatoire, et comme une majorité d'algorithmes du BAC il est posé dans le contexte d'une suite.

Jetons-y un coup d'oeil ce soir:

EXERCICE 4 : (5 points) ***Candidats N'AYANT PAS SUIVI l'enseignement de spécialité mathématiques***

On considère la suite numérique (v_n) définie pour tout entier naturel n par $\begin{cases} v_0 = 1 \\ v_{n+1} = \dfrac{9}{6 - v_n} \end{cases}$.

Partie A

1. On souhaite écrire un algorithme affichant, pour un entier naturel n donné, tous les termes de la suite, du rang 0 au rang n.
   Parmi les trois algorithmes suivants, un seul convient. Préciser lequel en justifiant la réponse.

Algorithme N°1

Variables :
v est un réel
i et n sont des entiers naturels

Début de l'algorithme :
Lire n
v prend la valeur 1
Pour i variant de 1 à n faire
v prend la valeur $\dfrac{9}{6-v}$
Fin pour
Afficher v
Fin algorithme.

Algorithme N°2

Variables :
v est un réel
i et n sont des entiers naturels

Début de l'algorithme :
Lire n
Pour i variant de 1 à n faire
v prend la valeur 1
Afficher v
v prend la valeur $\dfrac{9}{6-v}$
Fin pour
Fin algorithme.

Algorithme N°3

Variables :
v est un réel
i et n sont des entiers naturels

Début de l'algorithme :
Lire n
v prend la valeur 1
Pour i variant de 1 à n faire
Afficher v
v prend la valeur $\dfrac{9}{6-v}$
Fin pour
Afficher v
Fin algorithme.

2. Pour $n = 10$, on obtient l'affichage suivant :

1	1,800	2,143	2,333	2,455	2,538	2,600	2,647	2,684	2,714

Question A)1)
On veut donc un algorithme affichant tous les termes d'une suite v, du rang 0 au rang n.

On a la gentillesse dans ce sujet bien sympa de vous proposer 3 algorithmes déjà complets, et simplement de vous demander de choisir le bon.

L'algorithme 1 est à rejeter.
La seule instruction de sortie "Afficher v" arrive en effet après l'instruction "Fin pour". Elle est donc en dehors de la boucle et ne sera exécutée qu'une seule fois. Cet algorithme affichera dans tous les cas 1 seul terme et ne répond pas à la question posée.

L'algorithme 2 est également à rejeter.
L'instruction de sortie "Afficher v" fait suite à une instruction "v prend la valeur 1". Cet algorithme n'affiche donc que des 1 au lieu des valeurs des termes de la suite.

Par élimination, il ne reste donc que l'algorithme 3.

Notons que si vous avez du mal à lire/comprendre les algorithmes, il était également possible de trouver la réponse de façon empirique en utilisant la calculatrice graphique.

L'énoncé a la gentillesse (décidément) de donner en question A)2) ce qu'est censé afficher l'algorithme.

2. Pour $n = 10$ on obtient l'affichage suivant :

1	1,800	2,143	2,333	2,455	2,538	2,600	2,647	2,684	2,714

Il suffit donc de programmer les 3 algorithmes et de voir lequel produit le bon affichage.

Voici les 3 programmes pour calculatrices TI-82 à TI-84:

Pack et mise en page par Laurae, rédigé par critor sur le forum, administrateurs de TI-Planet.org

```
NORMAL FLOAT AUTO REAL RADIAN MP
PROGRAM:LIB213N1
:Prompt N
:1→V
:For(I,1,N)
:9/(6-V)→V
:End
:V
```

```
NORMAL FLOAT AUTO REAL RADIAN MP
PROGRAM:LIB213N2
:Prompt N
:1→V
:For(I,1,N)
:1→V
:Disp V
:9/(6-V)→V
:End
```

```
NORMAL FLOAT AUTO REAL RADIAN MP
PROGRAM:LIB213N3
:Prompt N
:1→V
:For(I,1,N)
:Disp V
:9/(6-V)→V
:End
:V
```

Voici maintenant leurs affichages respectifs:

```
NORMAL FLOAT AUTO REAL RADIAN MP
prgmLIB213N1
N=?6
                              2.6
```

```
NORMAL FLOAT AUTO REAL RADIAN MP
N=?6
                                1
                                1
                                1
                                1
                                1
                                1
                             Done
```

```
NORMAL FLOAT AUTO REAL RADIAN MP
N=?6
                                1
                              1.8
                      2.142857143
                      2.333333333
                      2.454545455
                      2.538461538
                              2.6
```

On constate bien les comportements décrits plus haut.
On peut y ajouter une petite précision: l'algorithme 1 affiche bien un seul terme, et il s'agit du terme de rang n.

Si vous êtes munis d'une Casio Graph/Prizm, vous pouvez réaliser la même chose en saisissant les programmes suivants:

```
LIB213N1
?→N↵
1→V↵
For 1→I To N↵
9÷(6-V)→V↵
Next↵
V
TOP BOTTOM SEARCH MENU A⇔a CHAR
```

```
LIB213N2
?→N↵
For 1→I To N↵
1→V↵
V◢
9÷(6-V)→V↵
Next
TOP BOTTOM SEARCH MENU A⇔a CHAR
```

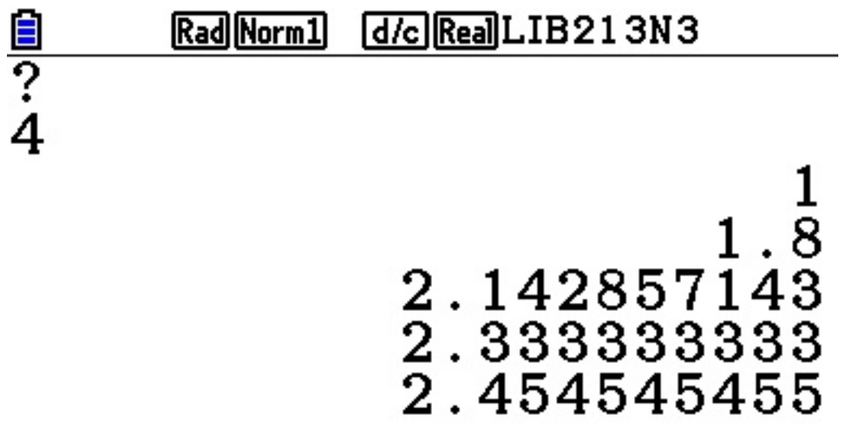
```
LIB213N3
?→N:1→V↵
For 1→I To N↵
V◢
9÷(6-V)→V↵
Next↵
V
CLEAR DISPLAY RELATNL I/O : ▷
```

Et voici leurs affichages:

```
Rad Norm1  d/c Real LIB213N1
?
4
                2.454545455
```

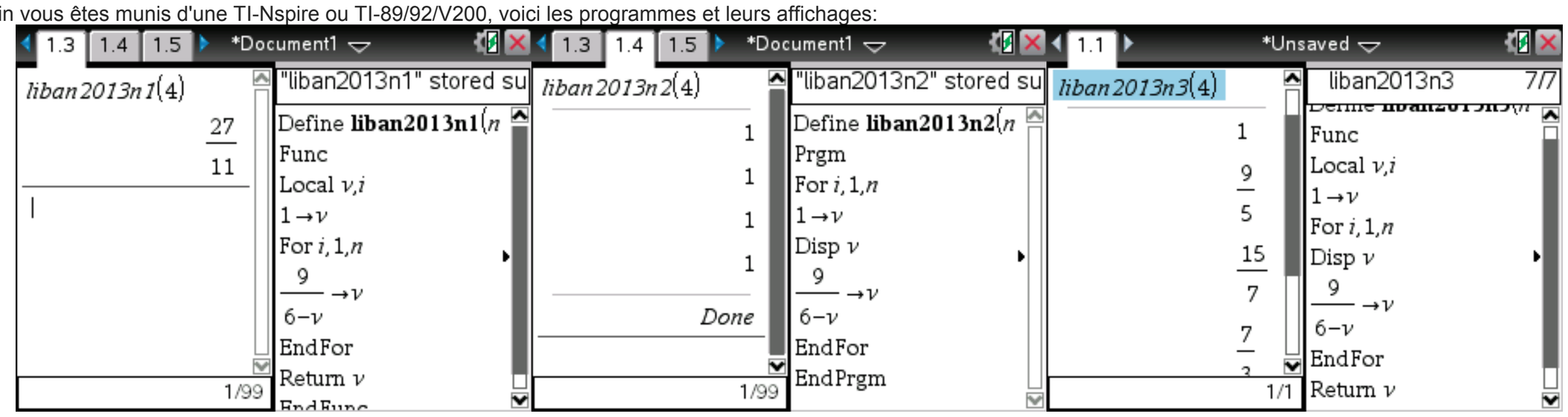
```
Rad Norm1  d/c Real LIB213N2
?
4
                              1
                              1
                              1
                              1
                            1.8
```

Pack et mise en page par Laurae, rédigé par critor sur le forum, administrateurs de TI-Planet.org

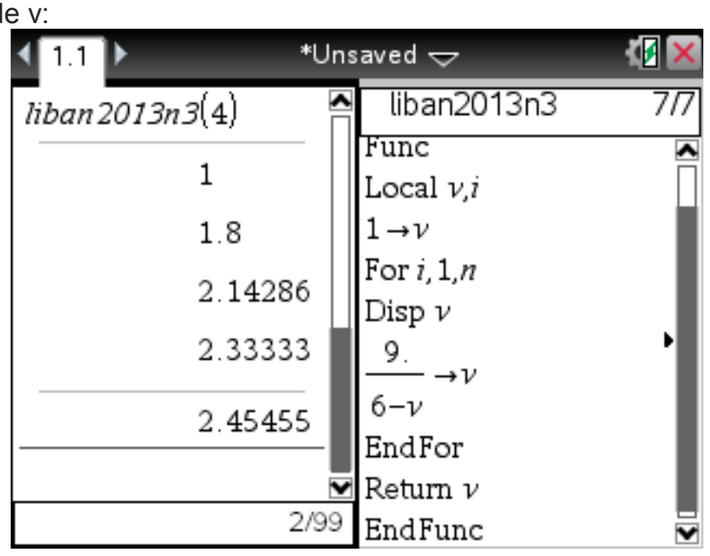


Pour le programme 2, l'affichage de 1.8 n'est déclenché par aucune instruction. C'est tout simplement la dernière valeur numérique utilisée dans le programme, que la calculatrice renvoie en tant que résultat, et notamment ici il s'agit de la dernière exécution de l'instruction d'affectation.

Si enfin vous êtes munis d'une TI-Nspire ou TI-89/92/V200, voici les programmes et leurs affichages:



On remarque que le programme 3 affiche des valeurs exactes fractionnaires pour v sur TI-Nspire CAS. Il suffit de provoquer le calcul de v en mode numérique et cela peut se faire par simple utilisation du séparateur décimal dans l'affectation de v:



Avant de nous quitter, posons-nous une dernière question au cas où...
Et si l'énoncé n'avait pas donné l'affichage correct de l'algorithme?

Il vous aurait suffi dans ce cas de définir la suite sur votre calculatrice et d'en demander un tableau de valeurs.

Sur TI-82 à TI-84, commencez par passer en mode 'suite' ou 'sequence' en anglais - quitter QUIT mode

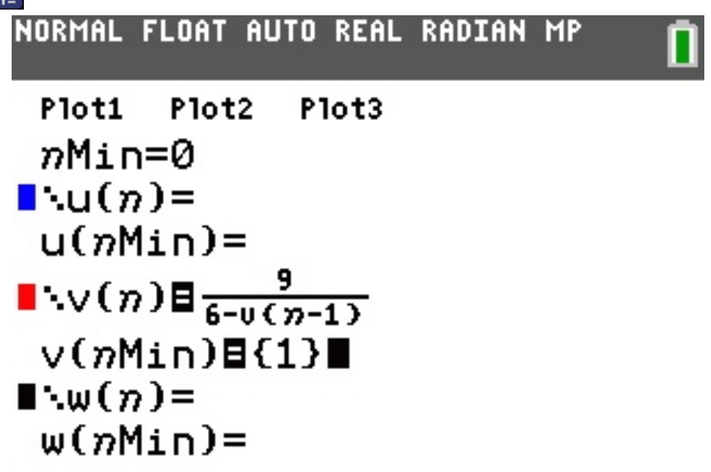
La suite vous est donc définie par une relation de récurrence $v_{n+1}=9/(6-v_n)$.

La calculatrice ne vous permet pas de définir le terme de rang n+1 mais uniquement le terme de rang n.

Vous devez donc commencer par réécrire cette relation au rang inférieur, c'est-à-dire en remplaçant tous les 'n' par des 'n-1'.

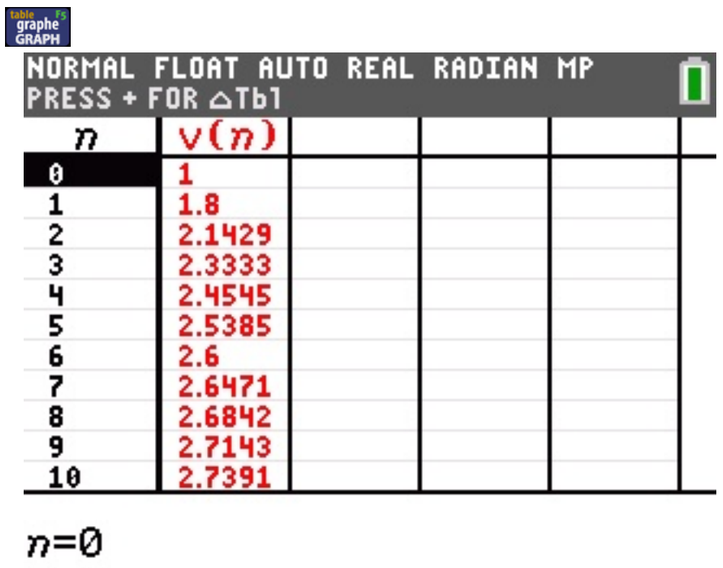
Cela nous donne $v_n=9/(6-v_{n-1})$

Une fois cette relation établie, il vous suffit d'aller la saisir -



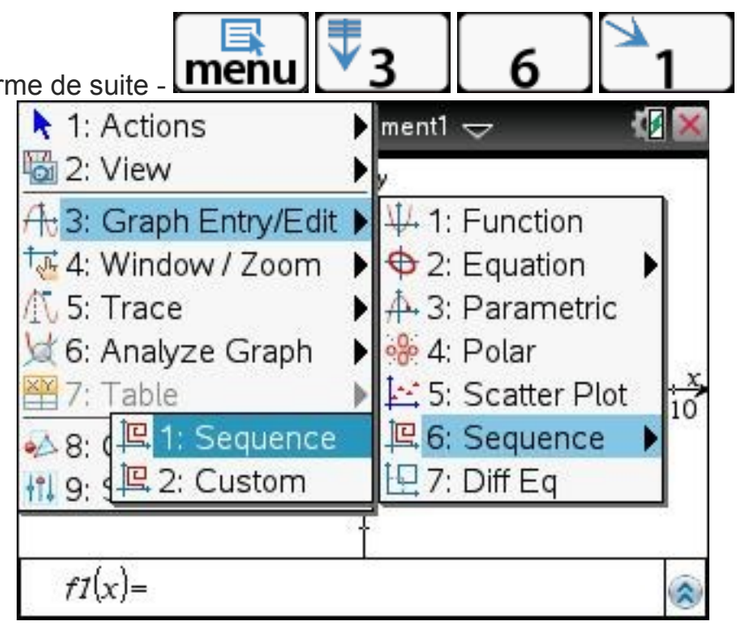
Selon l'état de votre calculatrice, il peut alors être nécessaire de modifier les paramètres du tableau de valeurs, à partir de 0 avec un pas de 1 dans le cas d'une suite -

Et vous pouvez enfin demander un tableau de valeurs -



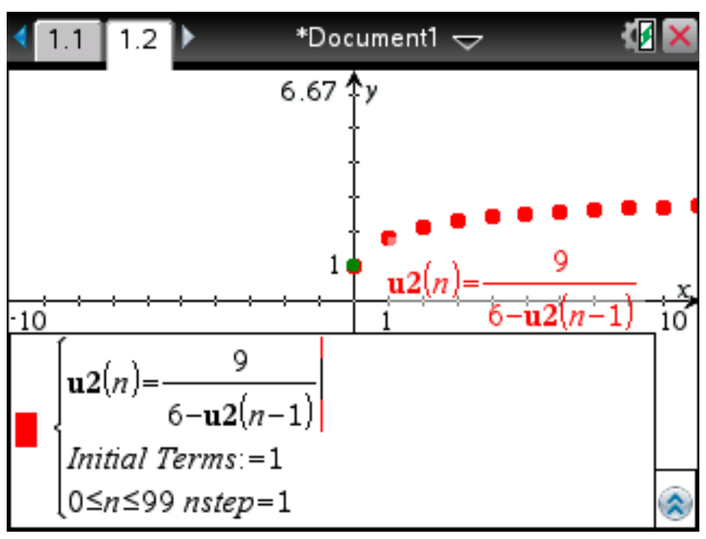
Sur TI-Nspire il vous faut utiliser la même relation.

Dans une application graphique, choisissez un entrée sous forme de suite -

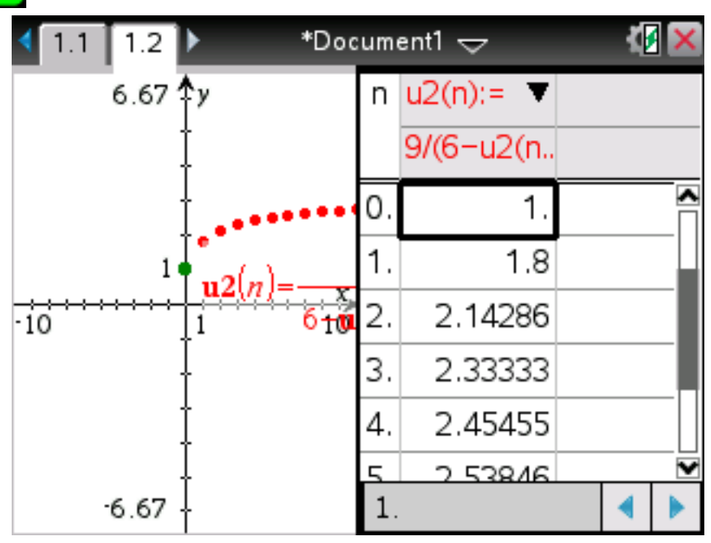


Saisissez alors la relation trouvée plus haut:

Pack et mise en page par Laurae, rédigé par critor sur le forum, administrateurs de TI-Planet.org

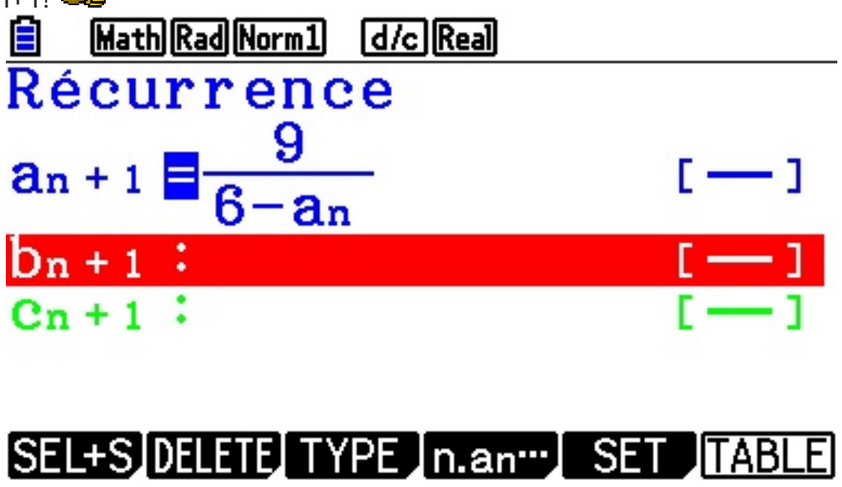


Et demandez enfin le tableau de valeurs - ctrl T

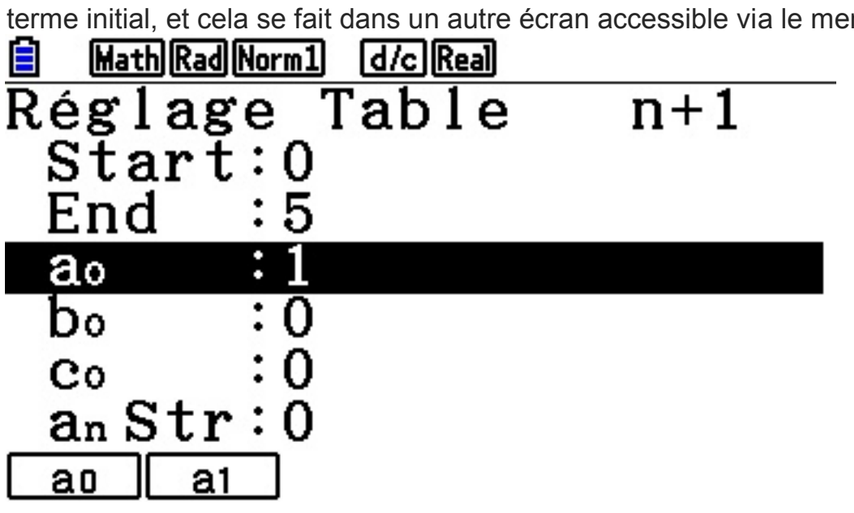


Sur une Casio Graph/Prizm, il suffit d'accéder à l'application 'Recurrence'.

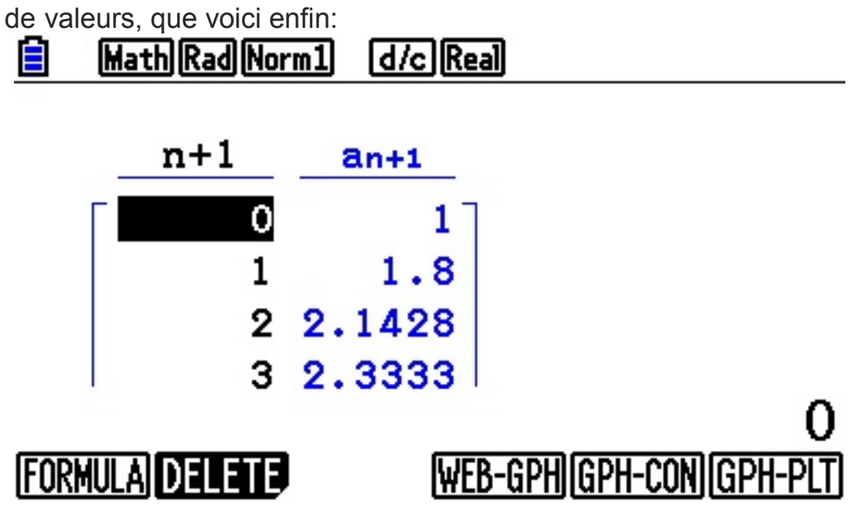
Il est possible ici d'y définir directement le terme de rang n+1!



Mais il ne faut surtout pas oublier de préciser la valeur du terme initial, et cela se fait dans un autre écran accessible via le menu [SET]:



On y précise également les paramètres de notre tableau de valeurs, que voici enfin:



Pack et mise en page par Laurae, rédigé par critor sur le forum, administrateurs de TI-Planet.org

Correction 27 : Correction algorithme Maths BAC ES/L 2013 (Liban)

de **critor** » 29 Mai 2013 18:46

Regardons un petit peu ce soir l'algorithme qui vient de tomber dans l'épreuve de Maths commune aux série ES Obligatoire et L Spécialité, pour les candidats passant le BAC 2013 dans les lycées français du Liban.

Encore une fois, comme dans une majorité des cas au BAC, l'algorithme tombe dans le contexte de suites:

Exercice 2 **5 points**

Commun à tous les candidats

Partie A

On considère la suite (u_n) définie par $u_0 = 10$ et pour tout entier naturel n,

$$u_{n+1} = 0,9u_n + 1,2.$$

1. On considère la suite (v_n) définie pour tout entier naturel n par $v_n = u_n - 12$.
   a. Démontrer que la suite (v_n) est une suite géométrique dont on précisera le premier terme et la raison.
   b. Exprimer v_n en fonction de n.
   c. En déduire que pour tout entier naturel n, $u_n = 12 - 2 \times 0,9n$.
2. Déterminer la limite de la suite (v_n) et en déduire celle de la suite (u_n).

Partie B

En 2012, la ville de Bellecité compte 10 milliers d'habitants. Les études démographiques sur les dernières années ont montré que chaque année :

- 10 % des habitants de la ville meurent ou déménagent dans une autre ville ;
- 1 200 personnes naissent ou emménagent dans cette ville.

1. Montrer que cette situation peut être modélisée par la suite (u_n) où u_n désigne le nombre de milliers d'habitants de la ville de Bellecité l'année $2012 + n$.
2. Un institut statistique décide d'utiliser un algorithme pour prévoir la population de la ville de Bellecité dans les années à venir.
   Recopier et compléter l'algorithme ci-dessous pour qu'il calcule la population de la ville de Bellecité l'année $2012 + n$.

```
VARIABLES
a, i, n.
INITIALISATION
Choisir n
a prend la valeur 10
TRAITEMENT
Pour i allant de 1 à n,
a prend la valeur ....

SORTIE
Afficher a
```

Question B)1)

Une ville comportant initialement à l'année d'indice 0 (2012) 10000 habitants perd 10% de sa population chaque année (c'est-à-dire qu'elle en garde 90%), mais gagne parallèlement 1200 personnes, soit 1,2 milliers.
On souhaite modéliser cette évolution à l'aide d'une suite u, donnant le nombre de milliers d'habitants.

La valeur initiale est donc u_0=10.
On traduit alors les deux autres informations de l'énoncé à l'aide de la relation de récurrence: u_{n+1}=0,9*u_n+1,2.

On remarque en effet que c'est exactement la suite étudiée en partie A de l'exercice.

Question B)2)

On souhaite donc avoir un algorithme calculant les termes u_n de la suite.
L'énoncé, bien gentil, nous donne un algorithme à trou.

La variable a joue le rôle du terme de la suite: il est bien initialisé à 10.
La variable i varie de 1 à n et est donc un compteur permettant de calculer par récurrence les termes des rangs 1 à n.

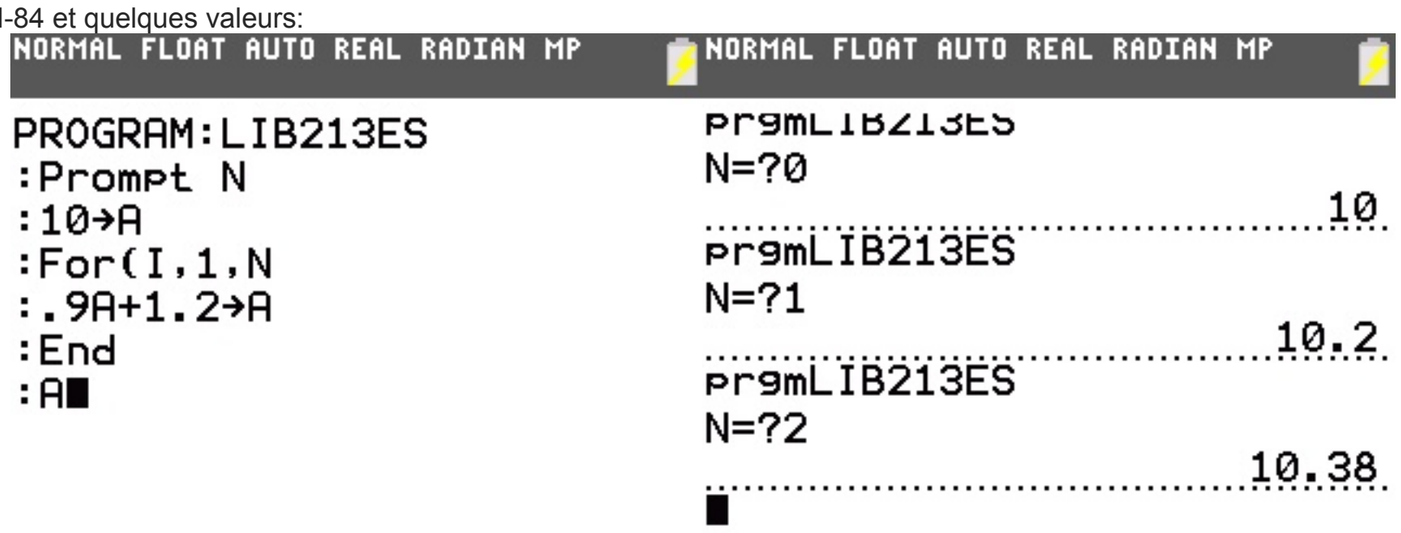
La seule information de l'énoncé non présente dans cet algorithme est la relation de récurrence et on peut donc le compléter de la façon suivante:

```
Choisir n
a prend la valeur 10
Pour i allant de 1 à n
   a prend la valeur 0,9*a+1,2
FinPour
```

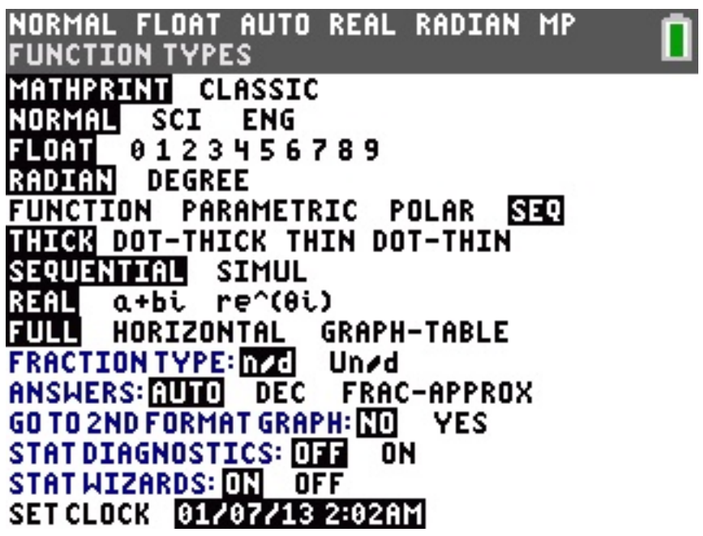
La calculatrice peut alors nous aider à vérifier si notre algorithme est correct.

Il suffit de le programmer et de comparer les valeurs affichées avec celles de la suite.

Voici le programme pour TI-82 à TI-84 et quelques valeurs:



Passons donc en mode 'suite' ou 'sequence' en anglais - 



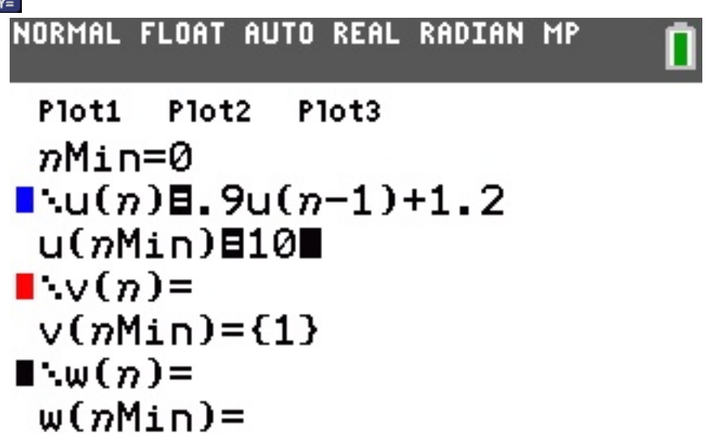
La suite vous est donc définie par une relation de récurrence $u_{n+1}=0,9*u_n+1,2$.

La calculatrice ne vous permet pas de définir le terme de rang n+1 mais uniquement le terme de rang n.

Vous devez donc commencer par réécrire cette relation un rang en-dessous, c'est-à-dire en remplaçant tous les 'n' par des 'n-1'.

Cela nous donne $u_n=0,9*u_{n-1}+1,2$

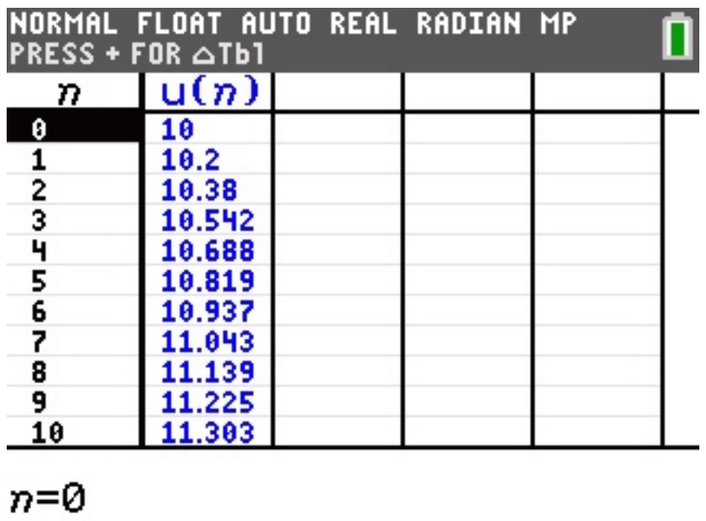
Une fois cette relation établie, il vous suffit d'aller la saisir - 



Selon l'état de votre calculatrice, il peut alors être nécessaire de modifier les paramètres du tableau de valeurs, à partir de 0 avec un pas de 1 dans le cas d'une suite - 

Et vous pouvez enfin demander le tableau de valeurs - 

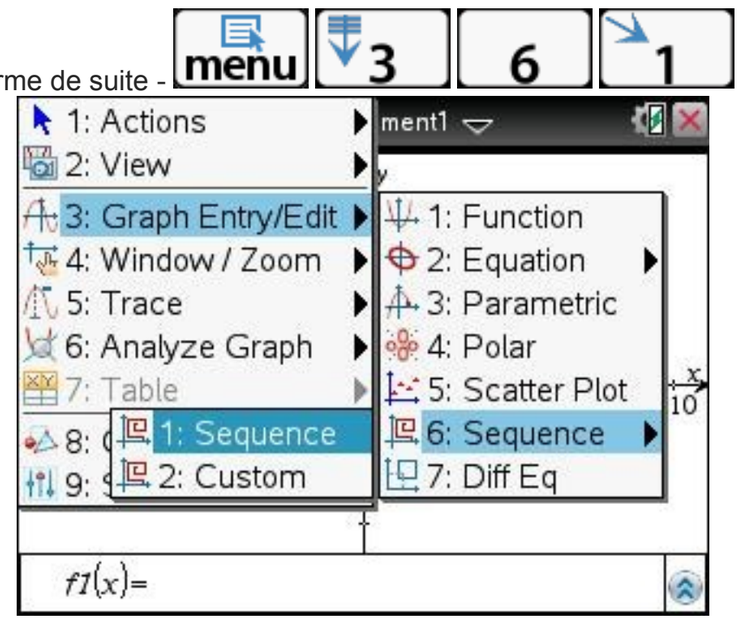
Pack et mise en page par Laurae, rédigé par critor sur le forum, administrateurs de TI-Planet.org



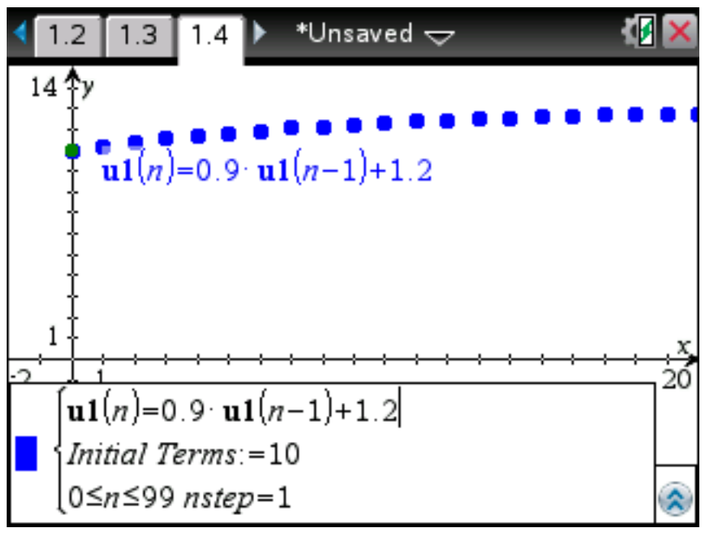
Notre programme et notre algorithme sont donc bons!

Sur TI-Nspire il vous faut utiliser la même relation.

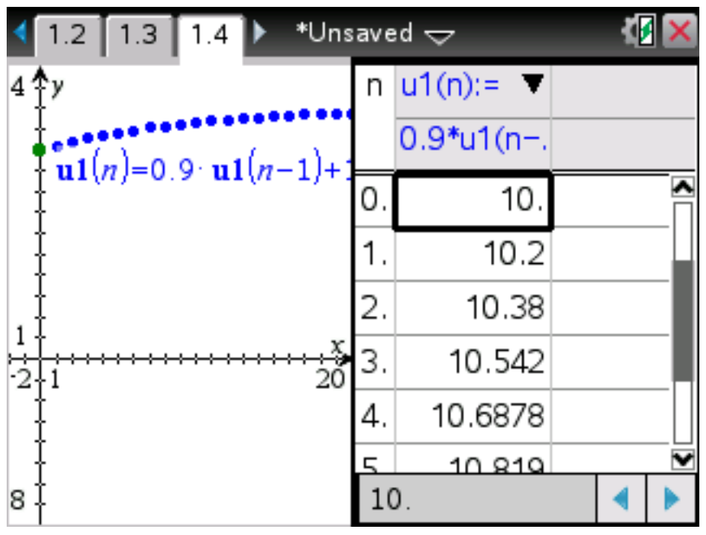
Dans une application graphique, choisissez un entrée sous forme de suite - menu 3 6 1



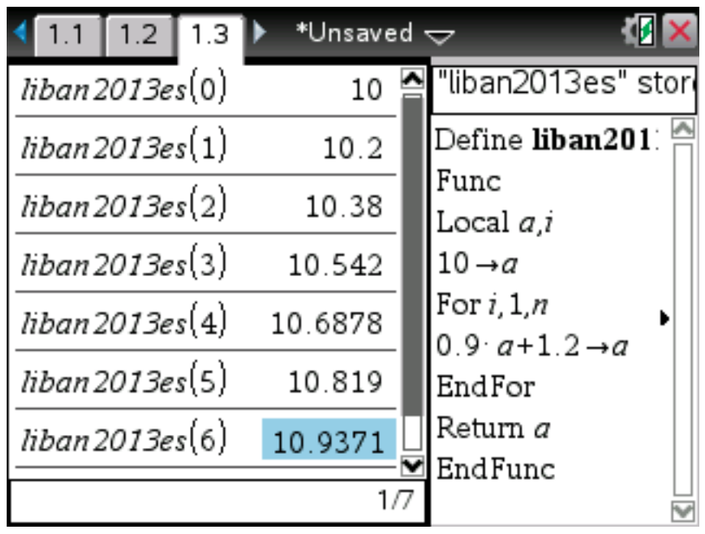
Saisissez alors la relation trouvée plus haut:



Et demandez le tableau de valeurs - ctrl T

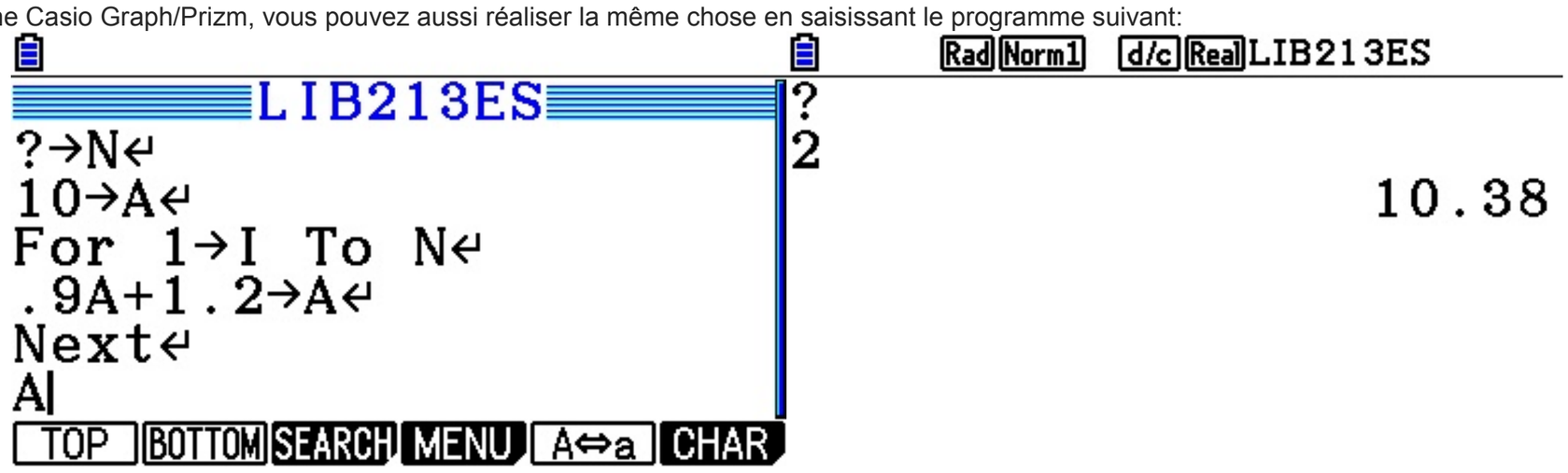


D'un autre côté, programmons l'algorithme et comparons les valeurs affichées:



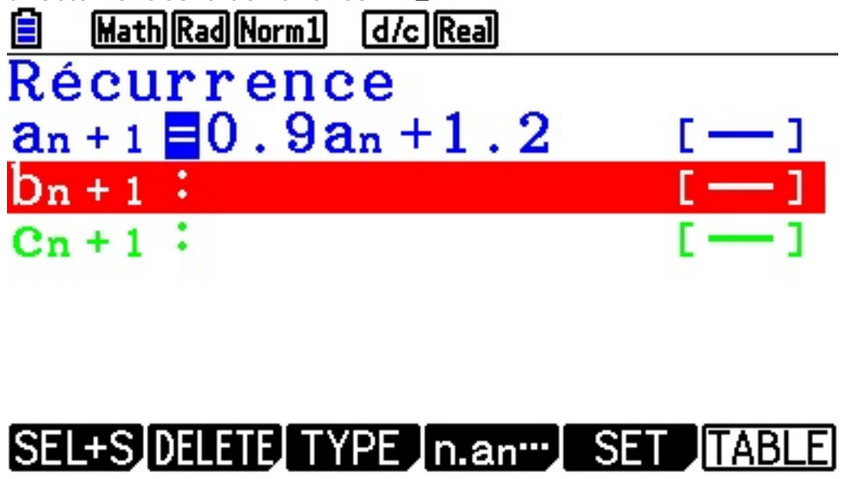
Encore une fois c'est correct.

Si vous êtes munis d'une Casio Graph/Prizm, vous pouvez aussi réaliser la même chose en saisissant le programme suivant:

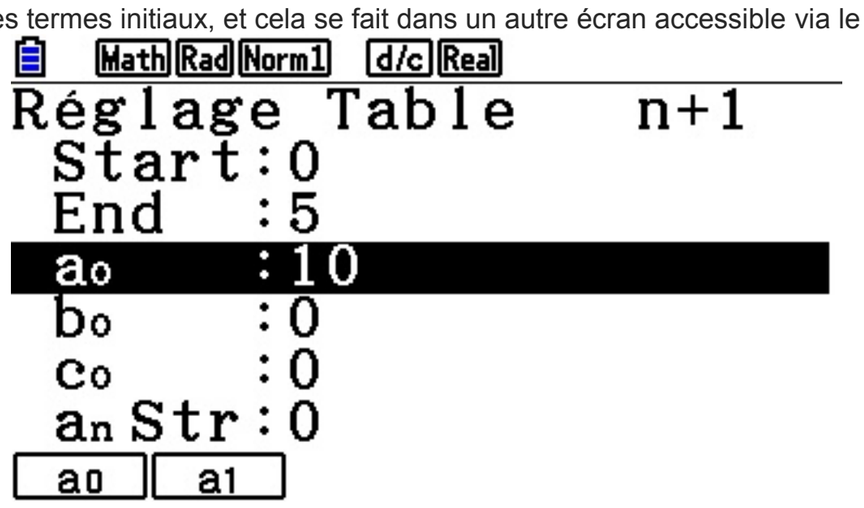


Il suffit maintenant d'accéder à l'application 'Récurrence' pour y définir notre suite.

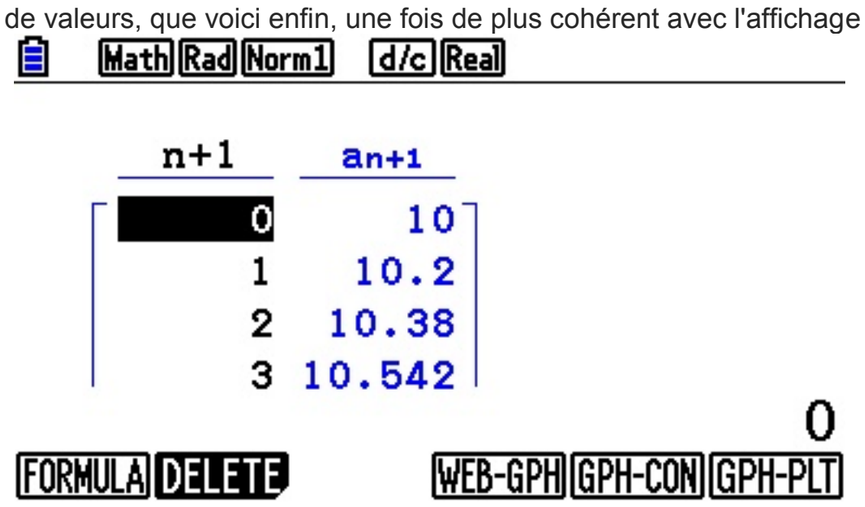
Ici, nul besoin de transformer l'expression - on peut saisir directement celle de l'énoncé!



Mais il ne faut surtout pas oublier de préciser la valeur des termes initiaux, et cela se fait dans un autre écran accessible via le menu [SET]:



On y précise également les paramètres de notre tableau de valeurs, que voici enfin, une fois de plus cohérent avec l'affichage du programme!



Pack et mise en page par Laurae, rédigé par critor sur le forum, administrateurs de TI-Planet.org

Correction 28 : Correction algorithme Spécialité BAC S 2013 (Amérique Nord)

de **critor** » 31 Mai 2013 01:40

En exercice 2 du sujet de Maths Spécialité qui vient de tomber aujourd'hui pour les candidats au BAC S 2013 dans les lycées français d'Amérique du Nord on a droit à de l'algorithmique, pour une fois non pas avec des suites mais avec de l'arithmétique!

Jetons-y un oeil ensemble ce soir:

Exercice 2 **5 points**
Candidats AYANT SUIVI l'enseignement de spécialité mathématiques

Partie A

On considère l'algorithme suivant :

Variables :	a est un entier naturel
	b est un entier naturel
	c est un entier naturel
Initialisation :	Affecter à c la valeur 0
	Demander la valeur de a
	Demander la valeur de b
Traitement :	Tant que $a > b$
	Affecter à c la valeur $c+1$
	Affecter à a la valeur $a-b$
	Fin de tant que
Sortie :	Afficher c
	Afficher a

1. Faire fonctionner cet algorithme avec $a = 13$ et $b = 4$ en indiquant les valeurs des variables à chaque étape.
2. Que permet de calculer cet algorithme ?

Partie B

À chaque lettre de l'alphabet, on associe, grâce au tableau ci-dessous, un nombre entier compris entre 0 et 25.

A	B	C	D	E	F	G	H	I	J	K	L	M
0	1	2	3	4	5	6	7	8	9	10	11	12

N	O	P	Q	R	S	T	U	V	W	X	Y	Z
13	14	15	16	17	18	19	20	21	22	23	24	25

On définit un procédé de codage de la façon suivante :

Étape 1 : À la lettre que l'on veut coder, on associe le nombre m correspondant dans le tableau.
Étape 2 : On calcule le reste de la division euclidienne de $9m+5$ par 26 et on le note p.
Étape 3 : Au nombre p, on associe la lettre correspondante dans le tableau.

1. Coder la lettre U.
2. Modifier l'algorithme de la partie A pour qu'à une valeur de m entrée par l'utilisateur, il affiche la valeur de p, calculée à l'aide du procédé de codage précédent.

Question A)1)
On nous demande donc de faire fonctionner l'algorithme et d'indiquer les valeurs de variables à chaque étape - c'est-à-dire de réaliser une trace de l'algorithme.

Une calculatrice peut ici grandement nous aider!
Prenons par exemple une TI-82 à TI-84, et commençons par y programmer l'algorithme original:

```
NORMAL FLOAT AUTO REAL RADIAN MP
PROGRAM:AMN2013S
:0→C
:Prompt A
:Prompt B
:While A>B
:C+1→C
:A-B→A
:End
:Disp C
:Disp A
```

L'algorithme utilise 3 variable a, b, c, ainsi qu'une boucle 'tant que'.

Pack et mise en page par Laurae, rédigé par critor sur le forum, administrateurs de TI-Planet.org

Nous allons modifier le programme afin d'afficher l'état des variables à chaque étape.
Il suffit en gros au choix:

- d'insérer un affichage des 3 variables en fin de boucle et avant la boucle
- d'insérer un affichage des 3 variables en début de boucle et après la boucle

C'est ce dernier choix que j'applique ici:

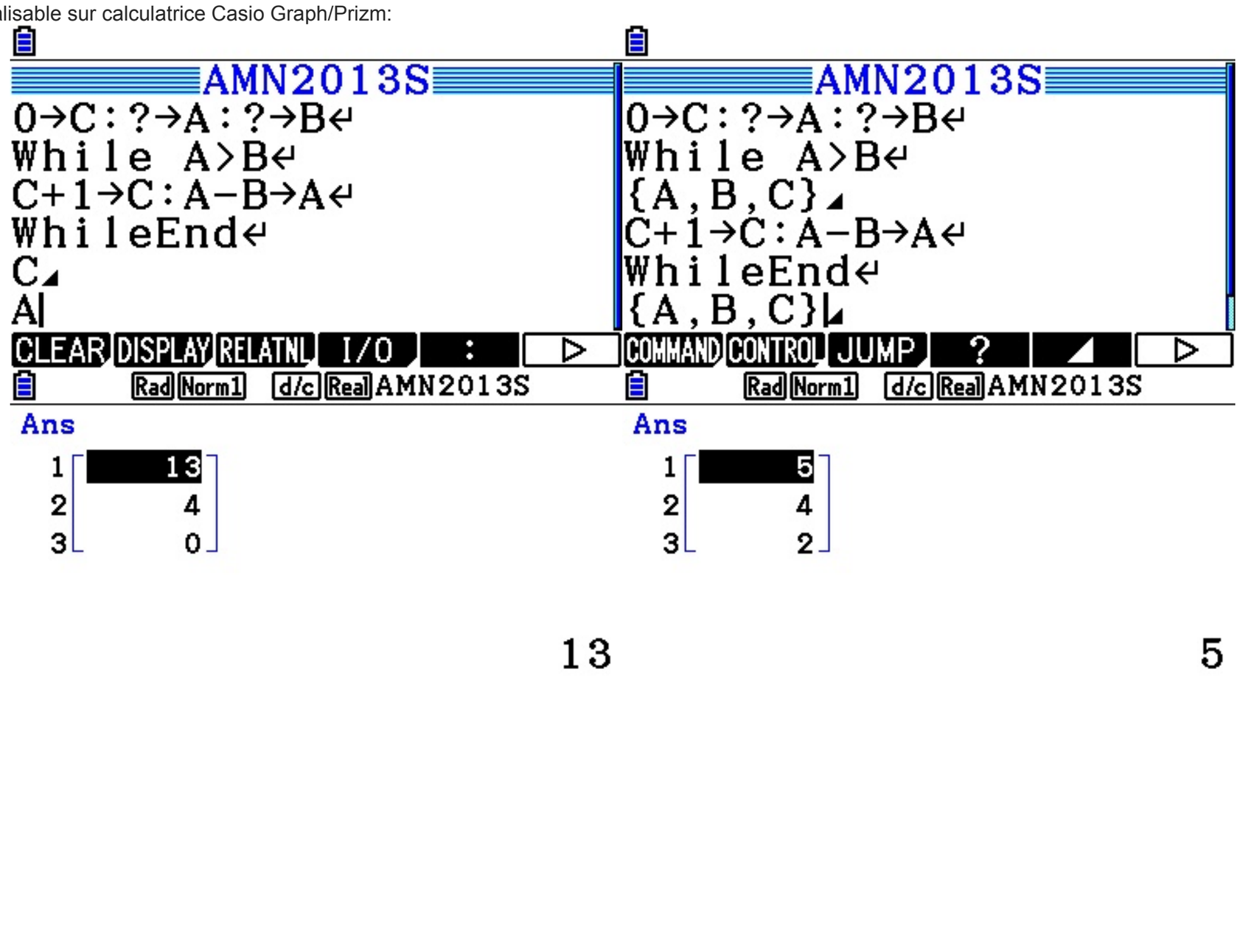
```
NORMAL FLOAT AUTO REAL RADIAN MP
PROGRAM:AMN2013S
:0→C
:Prompt A
:Prompt B
:While A>B
:Disp {A,B,C}
:C+1→C
:A-B→A
:End
:Disp {A,B,C}
```

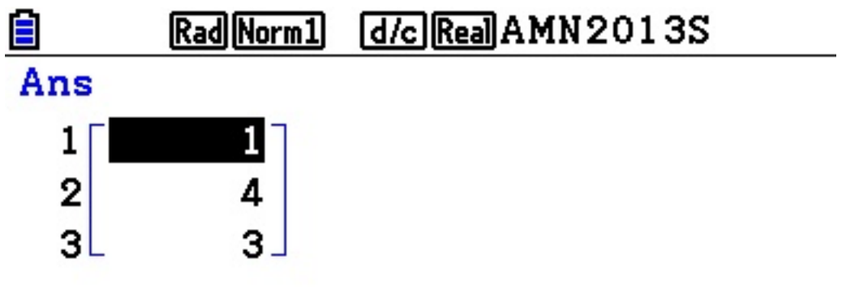
La calculatrice nous décrit alors toute seule la réponse qu'il nous suffit maintenant de mettre au propre:

```
NORMAL FLOAT AUTO REAL RADIAN MP
prgmAMN2013S
A=?13
B=?4
                    {13 4 0}
                     {9 4 1}
                     {5 4 2}
                     {1 4 3}
                        Done
```

Instruction	a	b	c
Affecter à c la valeur 0			0
Demander la valeur de a	13		0
Demander la valeur de b	13	4	0
Affecter à c la valeur c+1	13	4	1
Affecter à a la valeur a-b	9	4	1
Affecter à c la valeur c+1	9	4	2
Affecter à a la valeur a-b	5	4	2
Affecter à c la valeur c+1	5	4	3
Affecter à a la valeur a-b	1	4	3

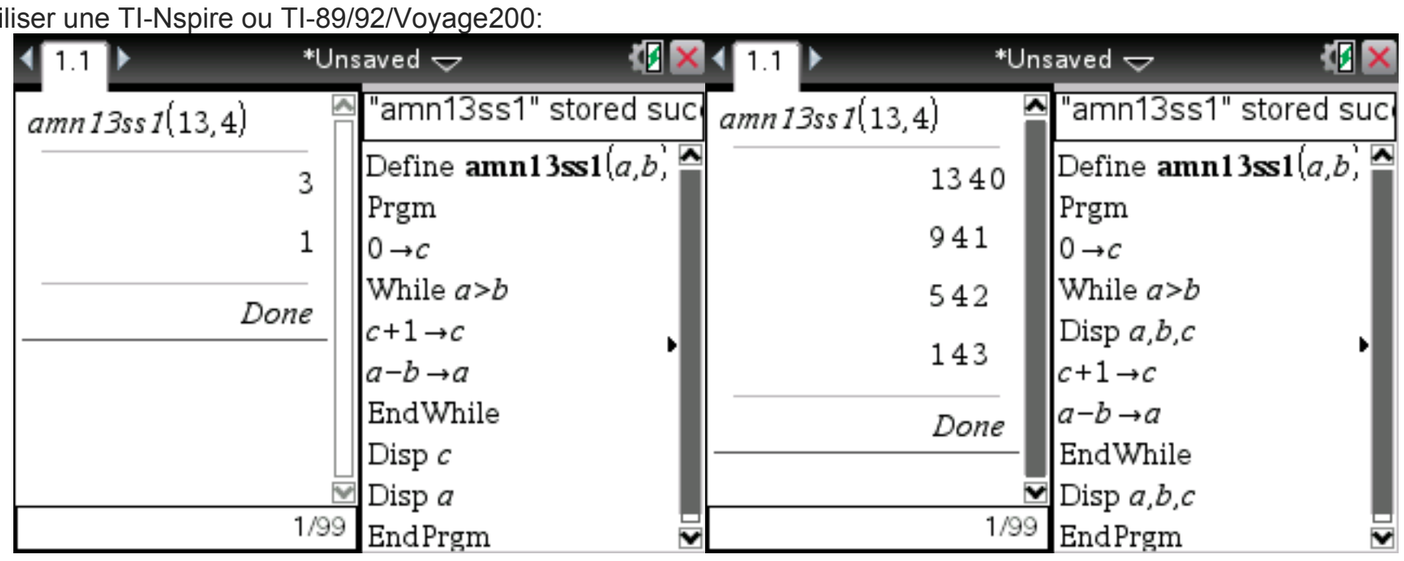
La même chose est réalisable sur calculatrice Casio Graph/Prizm:





1

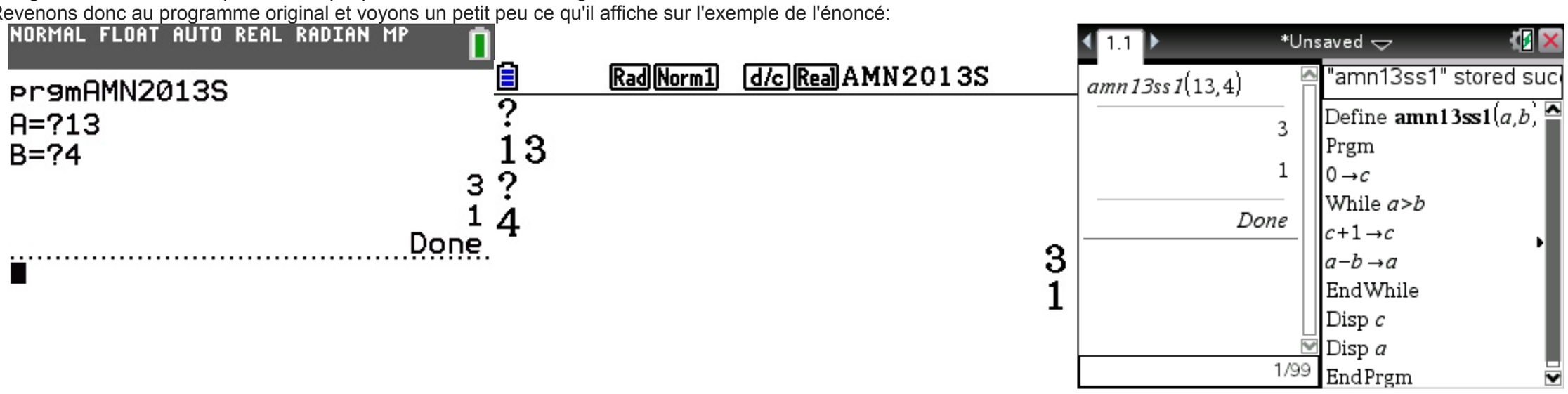
On peut aussi bien évidemment utiliser une TI-Nspire ou TI-89/92/Voyage200:



Question A)2)

Il s'agit maintenant de comprendre ce que permet de calculer cet algorithme.

Revenons donc au programme original et voyons un petit peu ce qu'il affiche sur l'exemple de l'énoncé:



3 et 1 sont respectivement les quotient et reste de la division euclidienne de 13 par 4, avec 13=4*3+1.
Cet algorithme permet donc de calculer les quotient et reste de la division euclidienne de a par b.

Quection B)1)

Nous souhaitons coder la lettre U.

- Etape 1: La lettre U est associée à m=20
- Etape 2: 9m+5=9*20+5=180+5=185 Or, 185=7*26+3. Donc, p=3.
- Etape 3: C'est la lettre C qui est associée à p=3

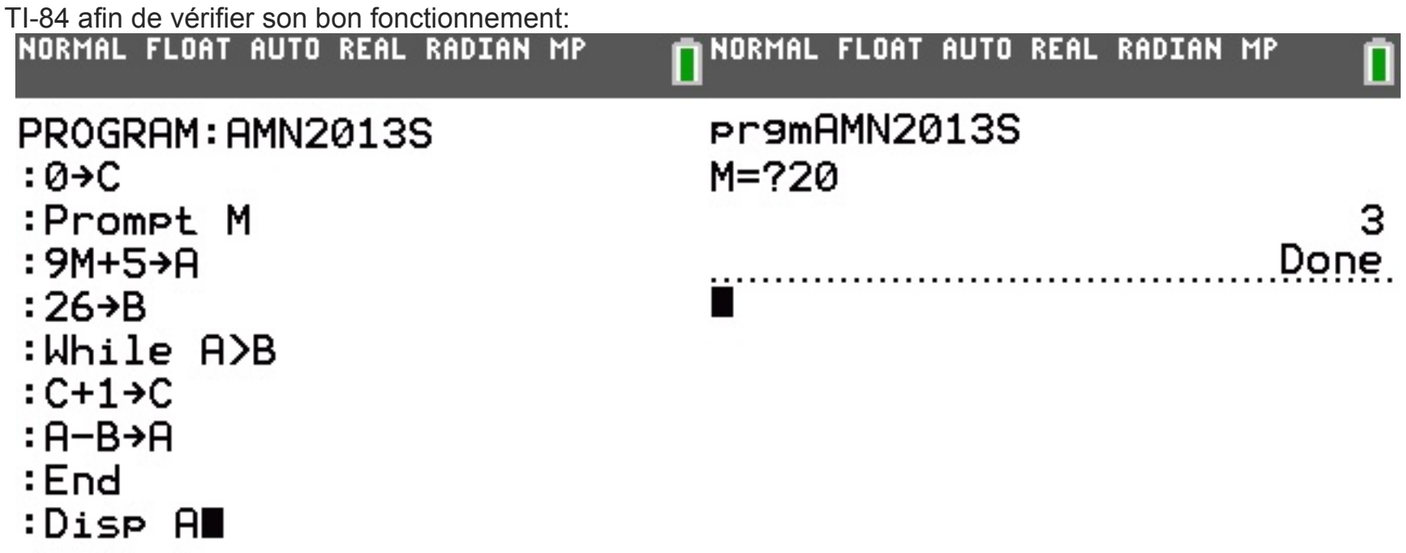
La lettre U une fois codée devient donc la lettre C.

Quection B)2)

Modifions donc l'algorithme afin d'effectuer directement ce codage, c'est-à-dire de donner le reste de la division euclidienne de 9m+5 par 26:

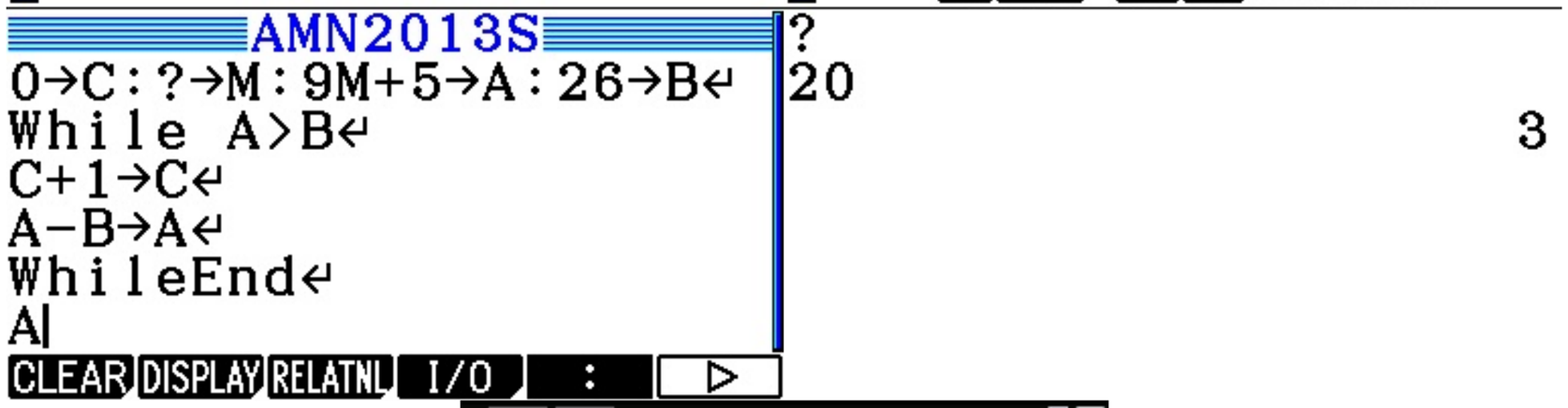
```
Variables:
   a est un entier naturel
   b est un entier naturel
   c est un entier naturel
*   m est un entier naturel
Initialisation:
   Affecter à c la valeur 0
*   Demander la valeur de m
*   Affecter à a la valeur 9*m+5
*   Affecter à b la valeur 26
Traitement:
   Tant que a>b
      Affecter à c la valeur c+1
      Affecter à a la valeur a-b
   Fin de tant que
Sortie:
*   Afficher a
```

Programmons-le sur notre TI-82 à TI-84 afin de vérifier son bon fonctionnement:



Le code 3 indique bien la lettre D - c'est bon!

On vérifie de même sur Casio Graph/Prizm ou TI-Nspire/89/92/Voyage200:



Correction 29 : Correction algorithme Obligatoire BAC S 2013 (Amérique Nord)

de **critor** » 31 Mai 2013 04:13

En exercice 2 du sujet de Maths Obligatoire qui vient de tomber aujourd'hui pour les candidats au BAC S 2013 dans les lycées français d'Amérique du Nord on a droit on pas à un mais **à deux algorithmes**, comme bien souvent dans le contexte de suite.

Jetons-y un oeil ensemble cette nuit:

Exercice 2 **5 points**

Candidats N'AYANT PAS SUIVI l'enseignement de spécialité mathématiques

On considère la suite (u_n) définie par $u_0 = 1$ et, pour tout entier naturel n,

$$u_{n+1} = \sqrt{2u_n}.$$

1. On considère l'algorithme suivant :

Variables :	n est un entier naturel
	u est un réel positif
Initialisation :	Demander la valeur de n
	Affecter à u la valeur 1
Traitement :	Pour i variant de 1 à n :
	| Affecter à u la valeur $\sqrt{2u}$
	Fin de Pour
Sortie :	Afficher u

a. Donner une valeur approchée à 10^{-4} près du résultat qu'affiche cet algorithme lorsque l'on choisit $n = 3$.

b. Que permet de calculer cet algorithme ?

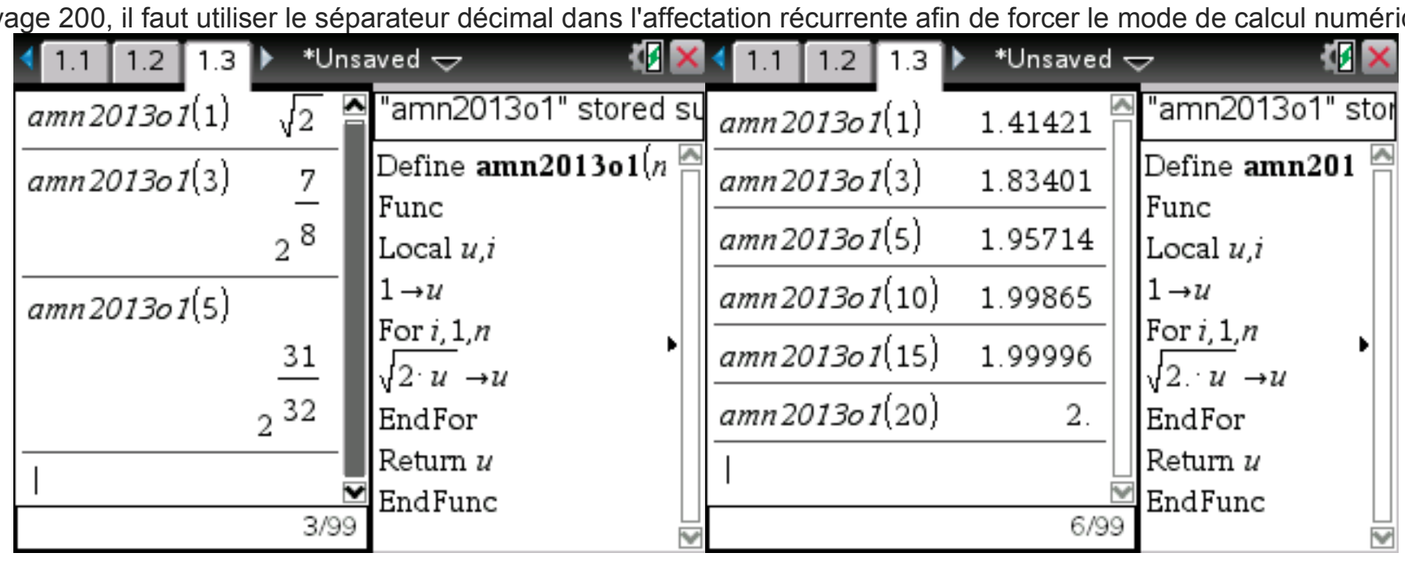
c. Le tableau ci-dessous donne des valeurs approchées obtenues à l'aide de cet algorithme pour certaines valeurs de n.

n	1	5	10	15	20
Valeur affichée	1,414 2	1,957 1	1,998 6	1,999 9	1,999 9

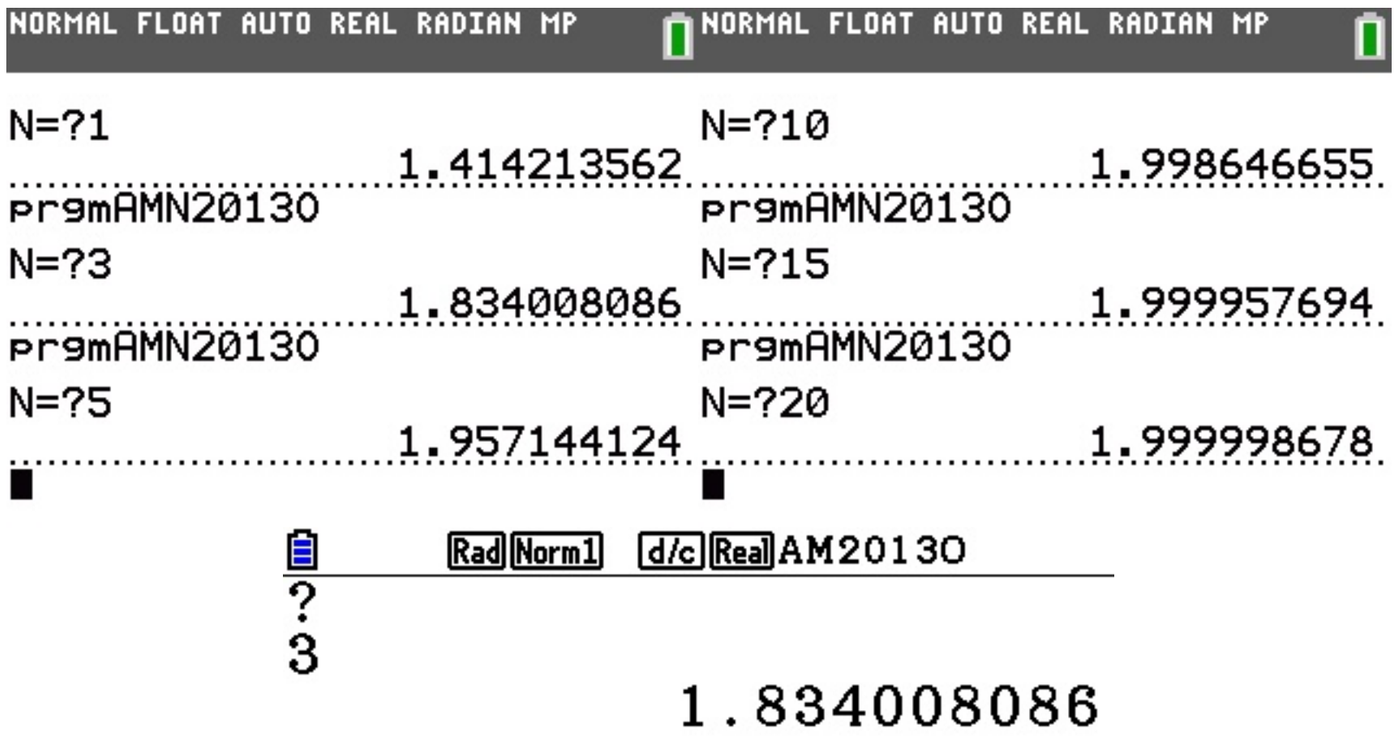
Question 1)a)
On nous demande donc ce qu'affiche l'algorithme lorsque n=3.

Demandons-le donc nous-même à notre tour à notre TI-82 à TI-84, Casio Graph Prizm:
Commençons par programmer l'algorithme:

Sur TI-Nspire CAS et TI-89/92/Voyage 200, il faut utiliser le séparateur décimal dans l'affectation récurrente afin de forcer le mode de calcul numérique approché:



Avons de répondre à la question, on peut avoir l'idée de vérifier le bon fonctionnement de l'algorithme avec les affichages que donnent gentillement la question 1)c)



La valeur est 1,8340 à 10^{-4} près.

Question 1)b)
On nous demande donc ce que calcule cet algorithme.
L'affectation récurrente "Affecter à u la valeur √(2u)" est la transcription exacte de la relation de récurrence $u_{n+1}=\sqrt{(2u_n)}$.
Lors de l'initialisation il y a bien une affectation "affecter à u la valeur 1" correspondant à $u_0=1$.

Cet algorithme calcule la valeur de u_n, le terme de rang n de la suite u.

Question 1)b)
La suite semble dans croissante et convergente vers 2.

Question 3)d)
Cet énoncé vous fait ensuite l'honneur d'un deuxième algorithme à la question 3)d):

d. Recopier l'algorithme ci-dessous et le compléter par les instructions du traitement et de la sortie, de façon à afficher en sortie la plus petite valeur de n telle que $u_n > 1{,}999$.

Variables :	n est un entier naturel
	u est un réel
Initialisation :	Affecter à n la valeur 0
	Affecter à u la valeur 1
Traitement :	
Sortie :	

Il s'agit donc de rechercher le rand du 1er terme de la suite u dépassant strictement 1,999.

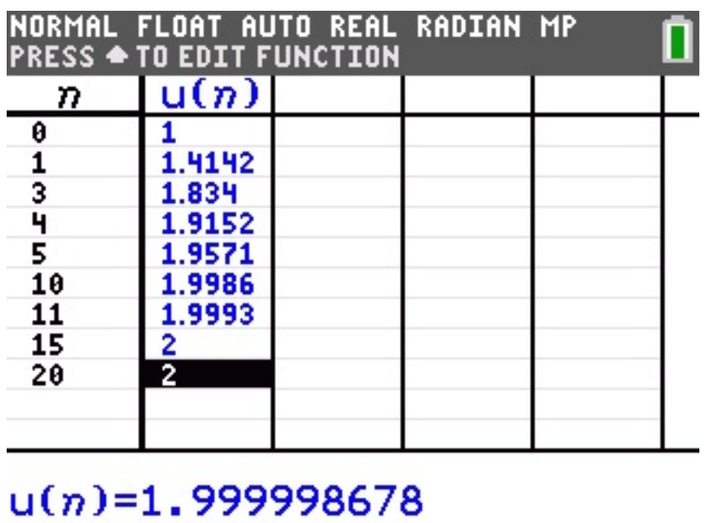
Comme on ne sait pas en combien d'essais on va le trouver, il nous faut donc une boucle 'tant que'.
La condition d'arrêt de cette boucle serait donc u>1,999, et la condition de poursuite de la boucle serait donc le contraire: u≤1,999.

Allons-y:
CODE: TOUT SÉLECTIONNER

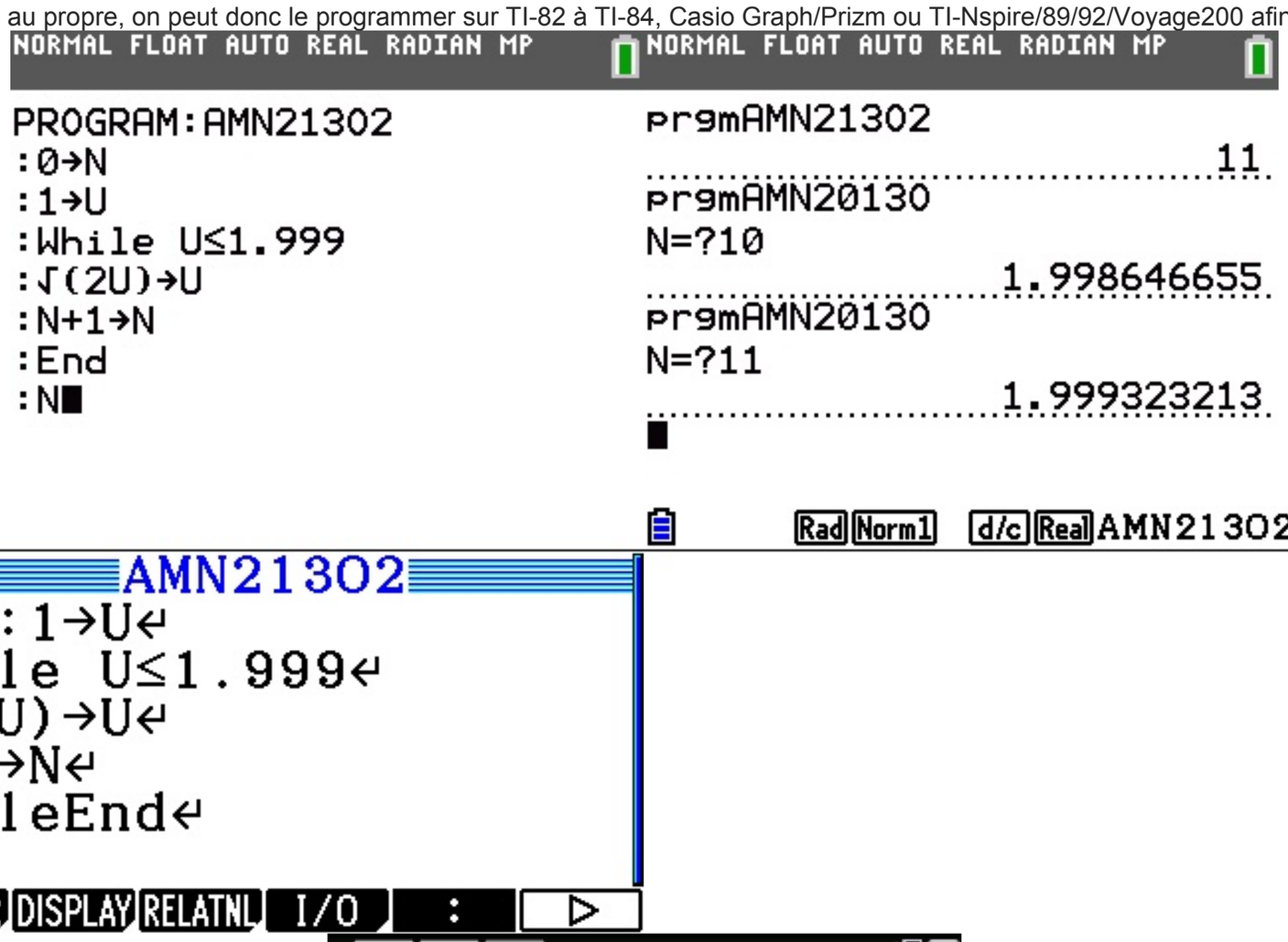
```
Variables:
   n est un entier naturel
   u est un réel
Initialisation:
   Affecter à n la valeur 0
   Affecter à u la valeur 1
Traitement:
   Tant que u≤1,999
      Affecter à u la valeur √(2u)
      Affecter à n la valeur n+1
   Fin du tant que
   Afficher n
```

Le tableau de valeurs de la calculatrice nous permet de connaître la réponse: n=11

Pack et mise en page par Laurae, rédigé par critor sur le forum, administrateurs de TI-Planet.org



Avant de recopier notre algorithme au propre, on peut donc le programmer sur TI-82 à TI-84, Casio Graph/Prizm ou TI-Nspire/89/92/Voyage200 afin de vérifier qu'il est bien exact:



Pack et mise en page par Laurae, rédigé par critor sur le forum, administrateurs de TI-Planet.org

Correction 30 : Correction algorithme Obligatoire BAC S 2013 (Amérique Nord)

de **critor** » 31 Mai 2013 04:13

En exercice 2 du sujet de Maths Obligatoire qui vient de tomber aujourd'hui pour les candidats au BAC S 2013 dans les lycées français d'Amérique du Nord on a droit on pas à un mais **à deux algorithmes**, comme bien souvent dans le contexte de suite.

Jetons-y un oeil ensemble cette nuit:

Exercice 2 **5 points**

Candidats N'AYANT PAS SUIVI l'enseignement de spécialité mathématiques

On considère la suite (u_n) définie par $u_0 = 1$ et, pour tout entier naturel n,

$$u_{n+1} = \sqrt{2u_n}.$$

1. On considère l'algorithme suivant :

Variables :	n est un entier naturel
	u est un réel positif
Initialisation :	Demander la valeur de n
	Affecter à u la valeur 1
Traitement :	Pour i variant de 1 à n :
	| Affecter à u la valeur $\sqrt{2u}$
	Fin de Pour
Sortie :	Afficher u

a. Donner une valeur approchée à 10^{-4} près du résultat qu'affiche cet algorithme lorsque l'on choisit $n = 3$.

b. Que permet de calculer cet algorithme ?

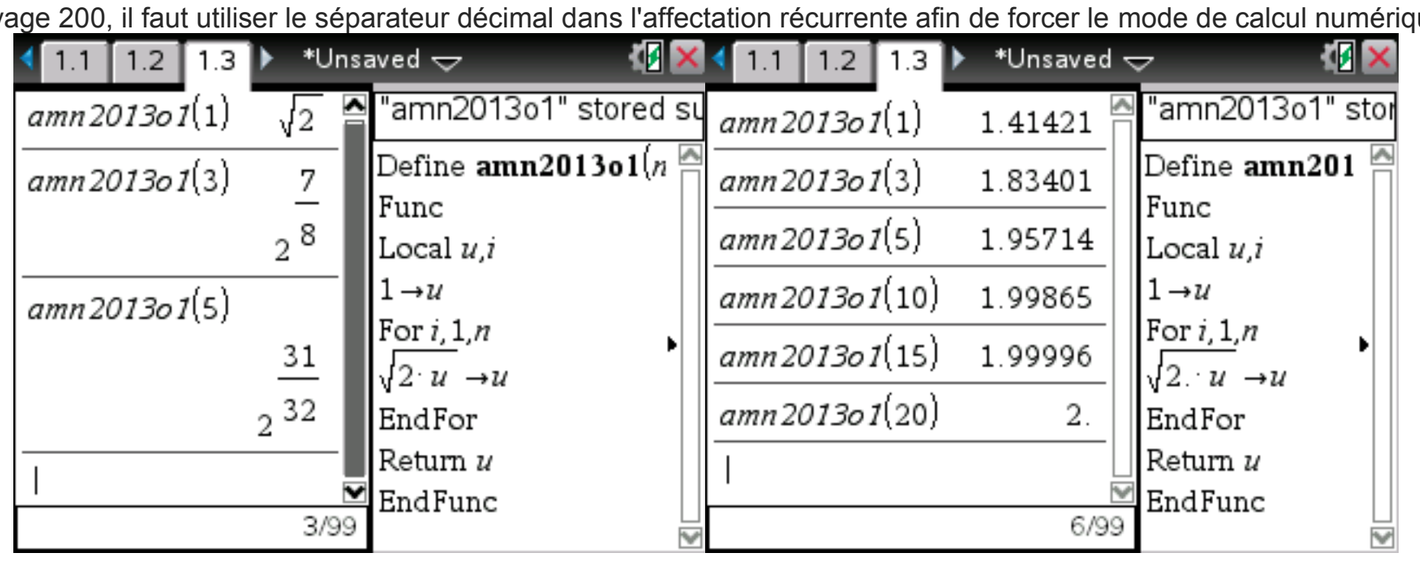
c. Le tableau ci-dessous donne des valeurs approchées obtenues à l'aide de cet algorithme pour certaines valeurs de n.

n	1	5	10	15	20
Valeur affichée	1,414 2	1,957 1	1,998 6	1,999 9	1,999 9

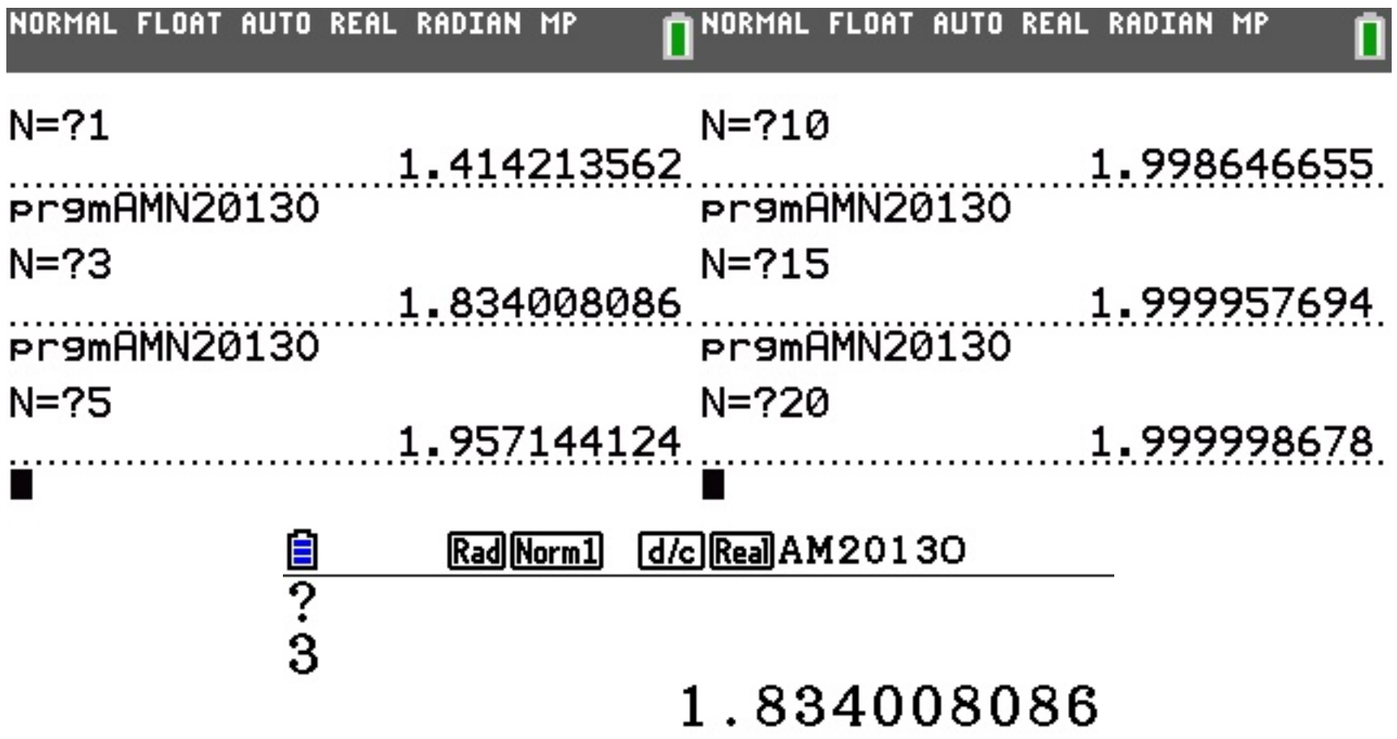
Question 1)a)
On nous demande donc ce qu'affiche l'algorithme lorsque n=3.

Demandons-le donc nous-même à notre tour à notre TI-82 à TI-84, Casio Graph Prizm:
Commençons par programmer l'algorithme:

Sur TI-Nspire CAS et TI-89/92/Voyage 200, il faut utiliser le séparateur décimal dans l'affectation récurrente afin de forcer le mode de calcul numérique approché:



Avons de répondre à la question, on peut avoir l'idée de vérifier le bon fonctionnement de l'algorithme avec les affichages que donnent gentillement la question 1)c)



La valeur est 1,8340 à 10^{-4} près.

Question 1)b)
On nous demande donc ce que calcule cet algorithme.
L'affectation récurrente "Affecter à u la valeur √(2u)" est la transcription exacte de la relation de récurrence $u_{n+1}=\sqrt{(2u_n)}$.
Lors de l'initialisation il y a bien une affectation "affecter à u la valeur 1" correspondant à $u_0=1$.

Cet algorithme calcule la valeur de u_n, le terme de rang n de la suite u.

Question 1)b)
La suite semble dans croissante et convergente vers 2.

Question 3)d)
Cet énoncé vous fait ensuite l'honneur d'un deuxième algorithme à la question 3)d):

d. Recopier l'algorithme ci-dessous et le compléter par les instructions du traitement et de la sortie, de façon à afficher en sortie la plus petite valeur de n telle que $u_n > 1{,}999$.

Variables :	n est un entier naturel
	u est un réel
Initialisation :	Affecter à n la valeur 0
	Affecter à u la valeur 1
Traitement :	
Sortie :	

Il s'agit donc de rechercher le rand du 1er terme de la suite u dépassant strictement 1,999.

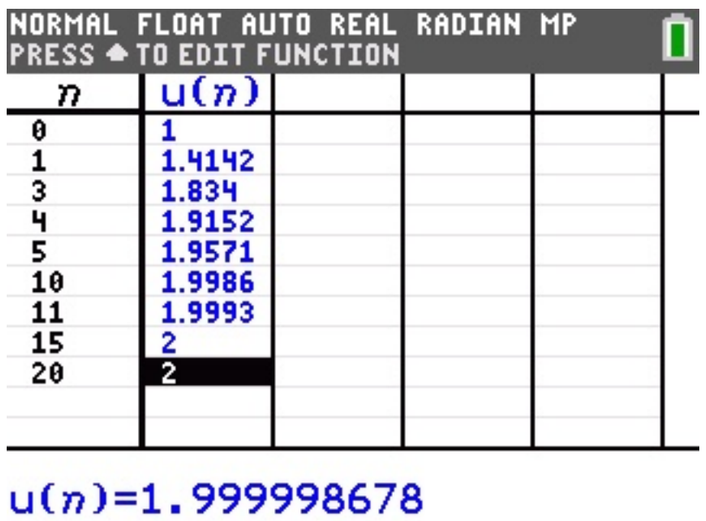
Comme on ne sait pas en combien d'essais on va le trouver, il nous faut donc une boucle 'tant que'.
La condition d'arrêt de cette boucle serait donc u>1,999, et la condition de poursuite de la boucle serait donc le contraire: u≤1,999.

Allons-y:

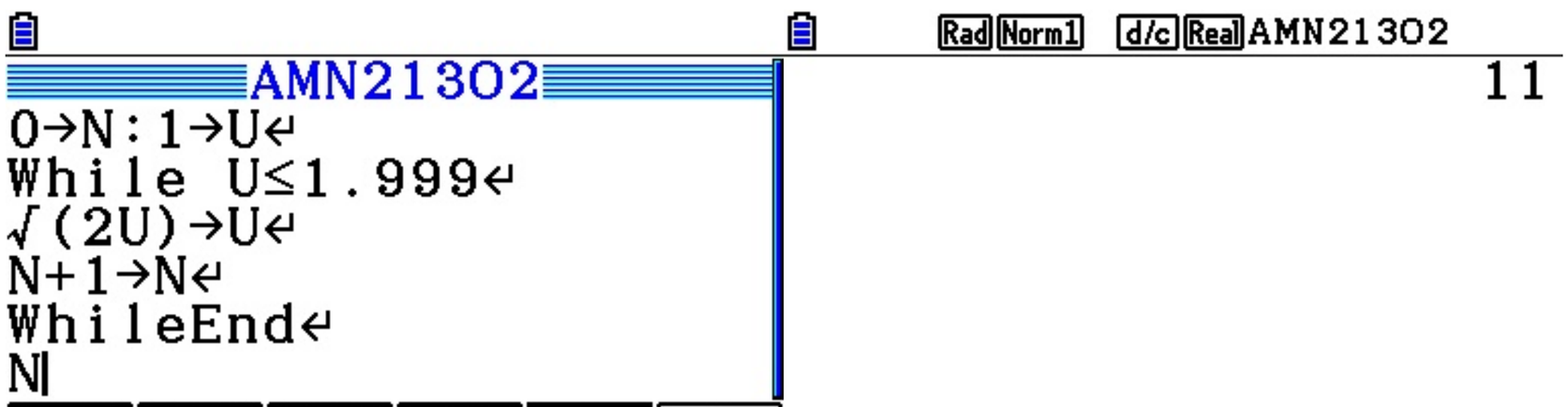
```
Variables:
   n est un entier naturel
   u est un réel
Initialisation:
   Affecter à n la valeur 0
   Affecter à u la valeur 1
Traitement:
   Tant que u≤1,999
      Affecter à u la valeur √(2u)
      Affecter à n la valeur n+1
   Fin du tant que
   Afficher n
```

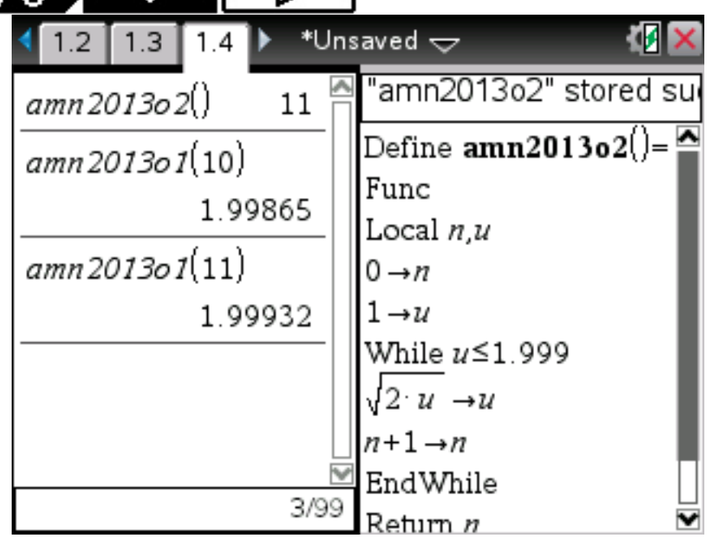
Le tableau de valeurs de la calculatrice nous permet de connaître la réponse: n=11

Pack et mise en page par Laurae, rédigé par critor sur le forum, administrateurs de TI-Planet.org



Avant de recopier notre algorithme au propre, on peut donc le programmer sur TI-82 à TI-84, Casio Graph/Prizm ou TI-Nspire/89/92/Voyage200 afin de vérifier qu'il est bien exact:





Correction 31 : Correction algorithme BAC ES/L 2013 (Amérique du Nord)

de **critor** » 31 Mai 2013 09:46

Regardons un petit peu ce matin l'algorithme qui vient de tomber dans l'épreuve de Maths commune aux séries ES Obligatoire et L Spécialité, pour les candidats passant les BAC ES et L 2013 dans les lycées français d'Amérique du Nord.

Encore une fois, comme dans une majorité des cas au BAC, l'algorithme tombe dans le contexte d'un exercice sur les suites:

EXERCICE 3 **5 points**

Candidats n'ayant pas suivi l'enseignement de spécialité

La bibliothèque municipale étant devenue trop petite, une commune a décidé d'ouvrir une médiathèque qui pourra contenir 100 000 ouvrages au total.
Pour l'ouverture prévue le 1er janvier 2013, la médiathèque dispose du stock de 35 000 ouvrages de l'ancienne bibliothèque augmenté de 7 000 ouvrages supplémentaires neufs offerts par la commune.

Partie A

Chaque année, la bibliothécaire est chargée de supprimer 5 % des ouvrages, trop vieux ou abîmés, et d'acheter 6 000 ouvrages neufs.
On appelle u_n le nombre, en milliers, d'ouvrages disponibles le 1er janvier de l'année $(2013+n)$.
On donne $u_0 = 42$.

1. Justifier que, pour tout entier naturel n , on a $u_{n+1} = u_n \times 0,95 + 6$.
2. On propose, ci-dessous, un algorithme, en langage naturel.
   Expliquer ce que permet de calculer cet algorithme.

```
Variables :
   U, N
Initialisation :
   Mettre 42 dans U
   Mettre 0 dans N
Traitement :
   Tant que U < 100
      U prend la valeur U × 0,95 + 6
      N prend la valeur N +1
   Fin du Tant que
Sortie
Afficher N.
```

3. À l'aide de votre calculatrice, déterminer le résultat obtenu grâce à cet algorithme.

Partie B

La commune doit finalement revoir ses dépenses à la baisse, elle ne pourra financer que 4 000 nouveaux ouvrages par an au lieu des 6 000 prévus.
On appelle v_n le nombre, en milliers, d'ouvrages disponibles le 1er janvier de l'année $(2013+n)$.

1. Identifier et écrire la ligne qu'il faut modifier dans l'algorithme pour prendre en compte ce changement.

Question A)2)
L'algorithme de l'énoncé comporte l'affectation récurrente "U prend la valeur U*0,95+6" correspondant à la relation de récurrence de la question précédente $u_{n+1}=u_n*0,95+6$.
De plus, on y trouve l'initialisation "Mettre 42 dans U" qui correspond au terme initial $u_0=42$.
Cet algorithme travaille donc sur la suite (u_n) de l'énoncé.

Mais que fait-il avec?
L'algorithme est principalement constitué d'une boucle 'tant que'.
Sa condition de poursuite est U<100, ce qui nous donne une condition d'arrêt terminant l'algorithme sur son contraire, U≥100.

L'algorithme recherche donc le rang du premier terme de la suite (u_n) supérieur ou égal à 100.

Dans le contexte de l'énoncé, il recherche donc le rang de l'année 2013+n à partir duquel la bibliothèque disposera d'au moins 100000 livres, la suite (u_n) exprimant en effet ce nombre de livres en milliers.

Question A)3)
On nous demande donc ce que répond l'algorithme. Il nous suffit pour cela de le programmer sur notre calculatrice graphique TI-82 à TI-84, Casio Graph/Prizm, ou TI-

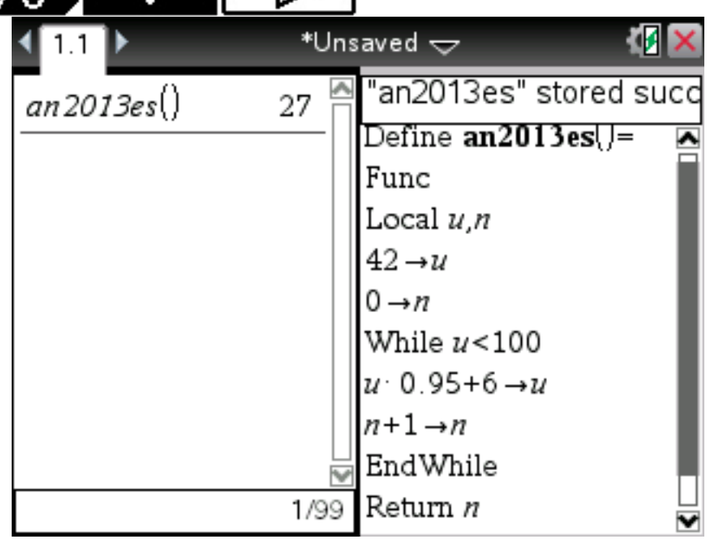
Nspire/89/92/Voyage200:

```
NORMAL FLOAT AUTO REAL RADIAN MP
PROGRAM:AN2013ES
:42→U
:0→N
:While U<100
:U*0.95+6→U
:N+1→N
:End
:N
```

```
NORMAL FLOAT AUTO REAL RADIAN MP
prgmAN2013ES
                                  27
```

```
AN2013ES
42→U:0→N↵
While U<100↵
U×0.95+6→U↵
N+1→N↵
WhileEnd↵
N
CLEAR DISPLAY RELATNL I/O :
```

```
Rad Norm1 d/c Real AN2013ES
                                  27
```



La réponse est donc n=27.
C'est donc en 2013+27=2040 que la bibliothèque comportera au moins 100000 ouvrages.

Si jamais vous n'étiez pas à l'aise avec la programmation de votre calculatrice ou tout simplement pour vérifier le résultat précédent, il était aussi possible de définir directement la suite dans la calculatrice et de demander son tableau de valeurs.

Prenons pour commencer une TI-82 à TI-84.

Passons donc en mode 'suite' ou 'sequence' en anglais - quitter QUIT mode

```
NORMAL FLOAT AUTO REAL RADIAN MP
FUNCTION TYPES
MATHPRINT CLASSIC
NORMAL SCI ENG
FLOAT 0 1 2 3 4 5 6 7 8 9
RADIAN DEGREE
FUNCTION PARAMETRIC POLAR SEQ
THICK DOT-THICK THIN DOT-THIN
SEQUENTIAL SIMUL
REAL a+bi re^(θi)
FULL HORIZONTAL GRAPH-TABLE
FRACTION TYPE: n/d Un/d
ANSWERS: AUTO DEC FRAC-APPROX
GO TO 2ND FORMAT GRAPH: NO YES
STAT DIAGNOSTICS: OFF ON
STAT WIZARDS: ON OFF
SET CLOCK 01/07/13 2:02AM
```

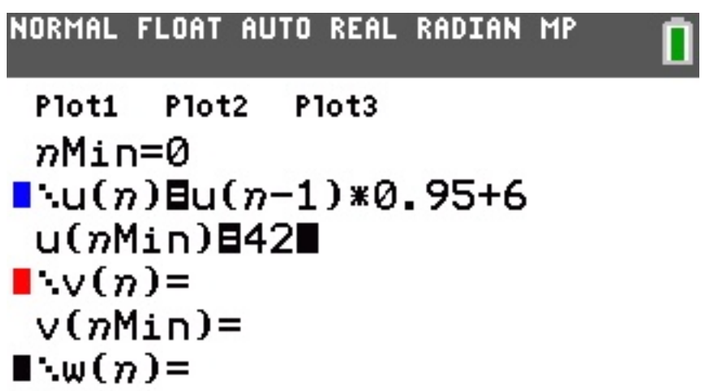
La suite vous est donc définie par une relation de récurrence $u_{n+1}=u_n*0,95+6$.

Mais la calculatrice ne vous permet pas de définir le terme de rang n+1 mais uniquement le terme de rang n.

Vous devez donc commencer par réécrire cette relation un rang en-dessous, c'est-à-dire en remplaçant tous les 'n' par des 'n-1'.
Cela nous donne $u_n=u_{n-1}*0,95+6$

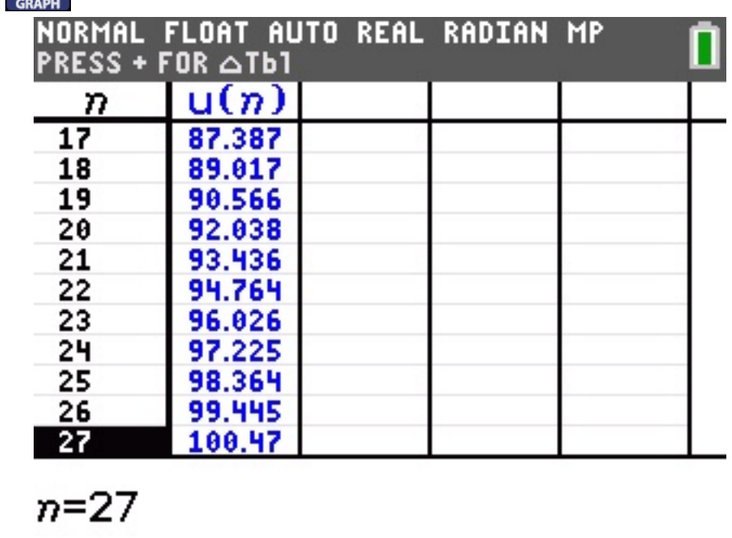
Une fois cette relation établie, il vous suffit d'aller la saisir - graph stats STAT PLOT F1 f(x) Y=

Pack et mise en page par Laurae, rédigé par critor sur le forum, administrateurs de TI-Planet.org



Selon l'état de votre calculatrice, il peut alors être nécessaire de modifier les paramètres du tableau de valeurs, à partir de 0 avec un pas de 1 dans le cas d'une suite - 2nde 2ND fenêtre WINDOW

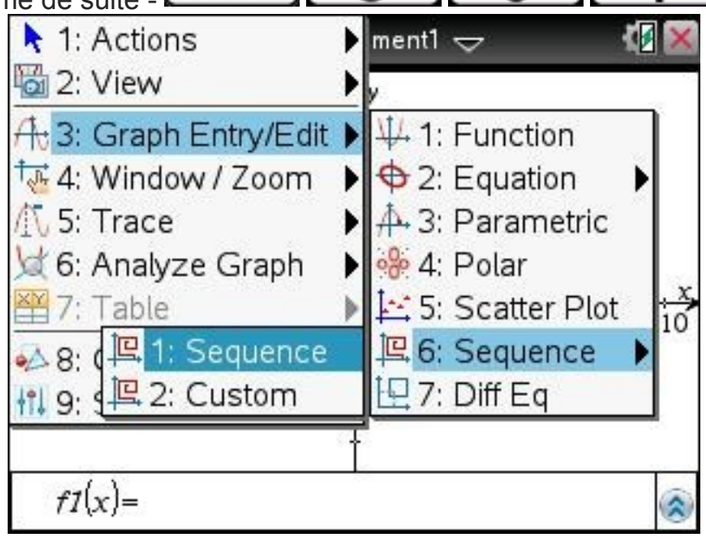
Et vous pouvez enfin demander le tableau de valeurs - 2nde 2ND table graphe GRAPH



La réponse est donc bien n=27.

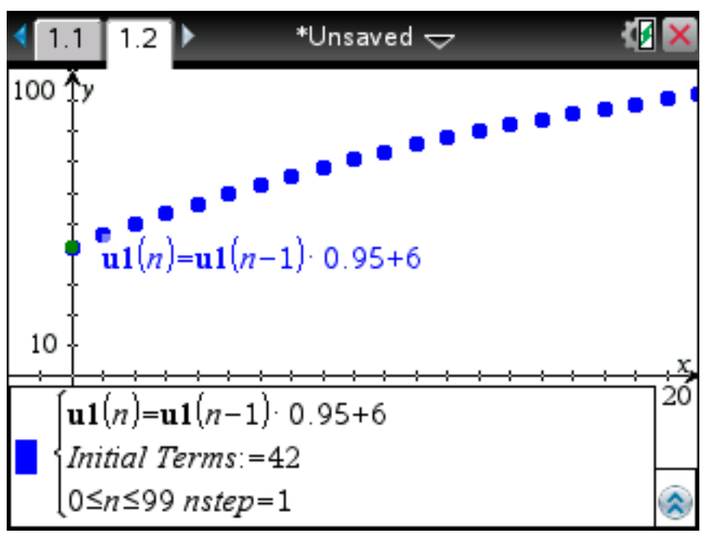
Sur TI-Nspire il vous faut utiliser la même relation.

Dans une application graphique, choisissez un entrée sous forme de suite - menu 3 6 1

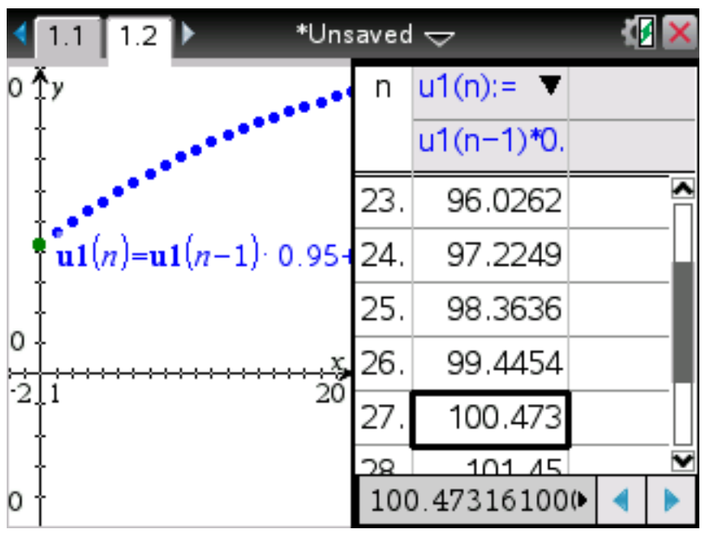


Saisissez alors la relation trouvée plus haut:

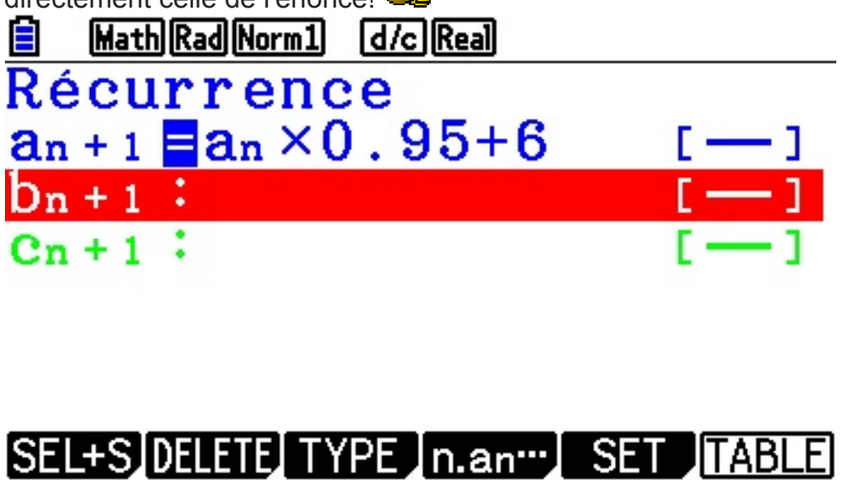
Pack et mise en page par Laurae, rédigé par critor sur le forum, administrateurs de TI-Planet.org



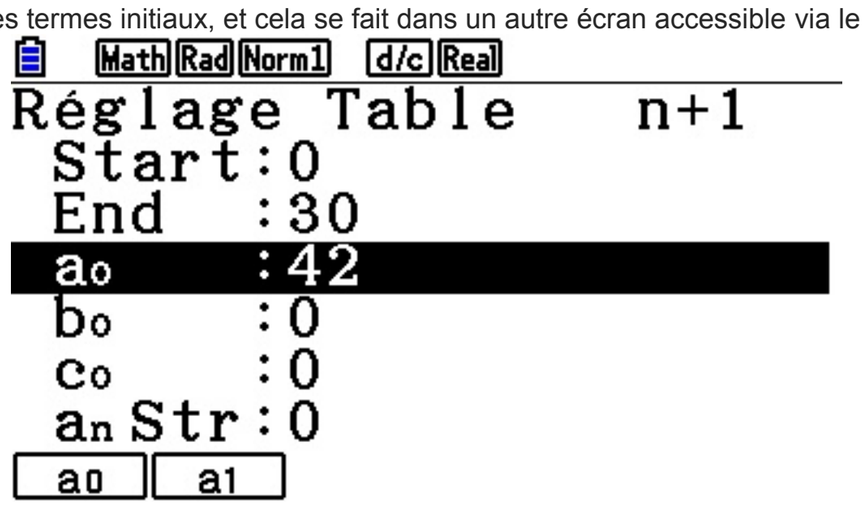
Et demandez le tableau de valeurs - ctrl T



Si vous êtes munis d'une Casio Graph/Prizm, il suffit d'accéder à l'application 'Récurrence' pour y définir notre suite.
Ici, nul besoin de transformer l'expression - on peut saisir directement celle de l'énoncé!

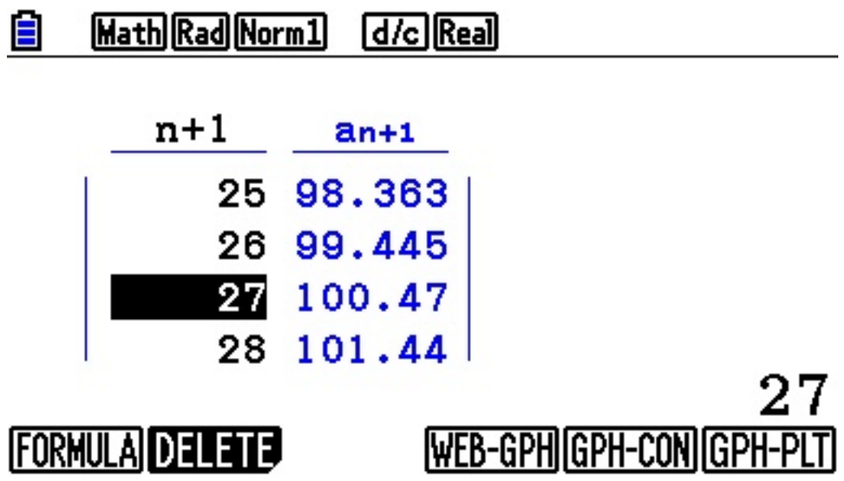


Mais il ne faut surtout pas oublier de préciser la valeur des termes initiaux, et cela se fait dans un autre écran accessible via le menu [SET]:



On y précise également les paramètres de notre tableau de valeurs, que voici enfin, une fois de plus cohérent avec l'affichage du programme!

Pack et mise en page par Laurae, rédigé par critor sur le forum, administrateurs de TI-Planet.org



Attention ici à ne pas se laisser induire en erreur par la légende du tableau.
La calculatrice affiche en effet que $a_{n+1} \approx 100,47$ lorsque n+1=27 soit n=26.
Mais cela veut donc dire que $a_{26+1} \approx 100,47$, soit $a_{27} \approx 100,47$, soit enfin bien $a_n \approx 100,47$ pour n=27.

Question B)1)
On modifie donc la suite puisque l'on n'ajoute plus que 4000 ouvrages chaque année au lieu de 6000.
La relation de récurrence $u_{n+1}=u_n*0,95+6$ devient donc $u_{n+1}=u_n*0,95+4$ (qui est en prime cadeau dans l'énoncé de la question B)2) suivante).
Nous allons donc modifier une seule ligne dans l'algorithme, l'affectation récurrente "U prend la valeur U*0,95+6" qui devient "U prend la valeur U*0,95+4".

A bientôt!

Correction 32 : Correction algorithme du sujet de Maths BAC S 2013 Polynésie

de **critor** » 08 Juin 2013 15:21

Aujourd'hui, nous allons regarder l'algorithme auquel ont eu droit sans surprise les candidats au BAC S 2013 de Polynésie française en exercice 1.

Il y a toutefois une petite originalité, puisque pour une fois l'algorithme ne tombe pas dans le contexte d'un exercice de suites mais de fonctions.

EXERCICE 1 : (6 points) ***Commun à tous les candidats***

On considère la fonction f définie sur $\mathbf{R}$ par $f(x) = (x+2)\mathrm{e}^{-x}$. On note $\mathscr{C}$ la courbe représentative de la fonction f dans un repère orthogonal.

1. Étude de la fonction f.

a. Déterminer les coordonnées des points d'intersection de la courbe $\mathscr{C}$ avec les axes du repère.

b. Étudier les limites de la fonction f en $-\infty$ et en $+\infty$. En déduire les éventuelles asymptotes à la courbe $\mathscr{C}$.

c. Étudier les variations de la fonction f sur $\mathbf{R}$.

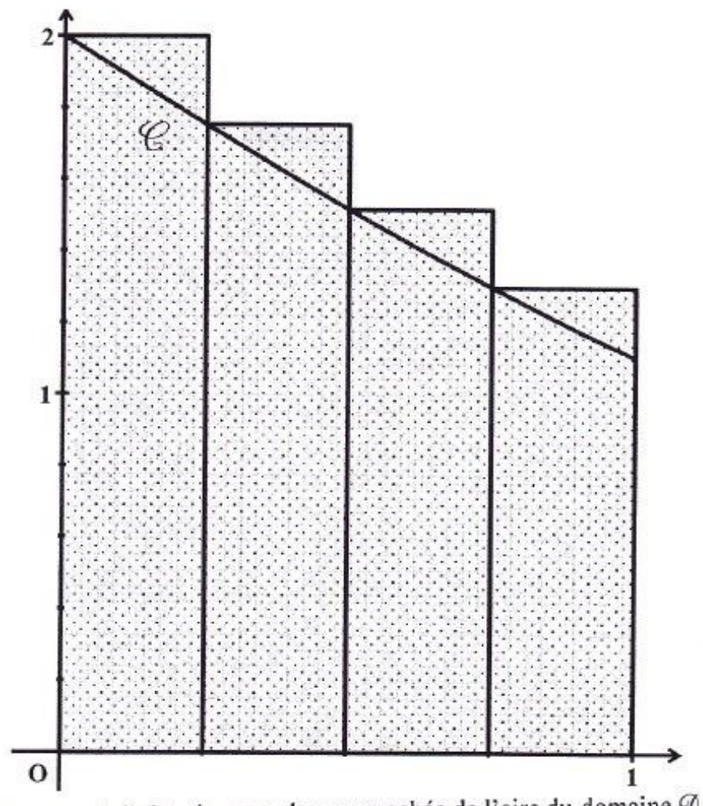
2. Calcul d'une valeur approchée de l'aire sous une courbe.

On note $\mathscr{D}$ le domaine compris entre l'axe des abscisses, la courbe $\mathscr{C}$ et les droites d'équation $x = 0$ et $x = 1$. On approche l'aire du domaine $\mathscr{D}$ en calculant une somme d'aires de rectangles.

a. Dans cette question, on découpe l'intervalle $[0,1]$ en quatre intervalles de même longueur :

- sur l'intervalle $\left[0,\frac{1}{4}\right]$, on construit un rectangle de hauteur $f(0)$
- sur l'intervalle $\left[\frac{1}{4},\frac{1}{2}\right]$, on construit un rectangle de hauteur $f\left(\frac{1}{4}\right)$
- sur l'intervalle $\left[\frac{1}{2},\frac{3}{4}\right]$, on construit un rectangle de hauteur $f\left(\frac{1}{2}\right)$
- sur l'intervalle $\left[\frac{3}{4},1\right]$, on construit un rectangle de hauteur $f\left(\frac{3}{4}\right)$.

Cette construction est illustrée ci-dessous.



L'algorithme ci-dessous permet d'obtenir une valeur approchée de l'aire du domaine $\mathscr{D}$ en ajoutant les aires des quatre rectangles précédents :

Variables :	k est un nombre entier
	S est un nombre réel
Initialisation :	Affecter à S la valeur 0
Traitement :	Pour k variant de 0 à 3
	Affecter à S la valeur $S+\frac{1}{4}f\left(\frac{k}{4}\right)$
	Fin Pour
Sortie :	Afficher S

Donner une valeur approchée à 10^{-3} près du résultat affiché par cet algorithme.

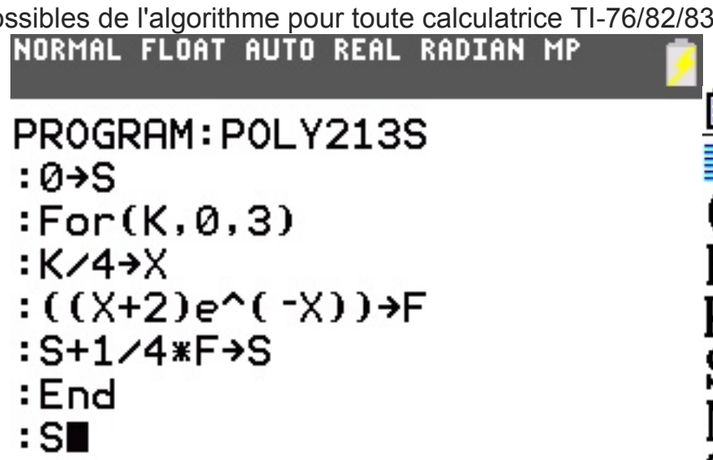
b. Dans cette question, N est un nombre entier strictement supérieur à 1. On découpe l'intervalle $[0,1]$ en N intervalles de même longueur. Sur chacun de ces intervalles, on construit un rectangle en procédant de la même manière qu'à la question 2.a.
Modifier l'algorithme précédent afin qu'il affiche en sortie la somme des aires des N rectangles ainsi construits.

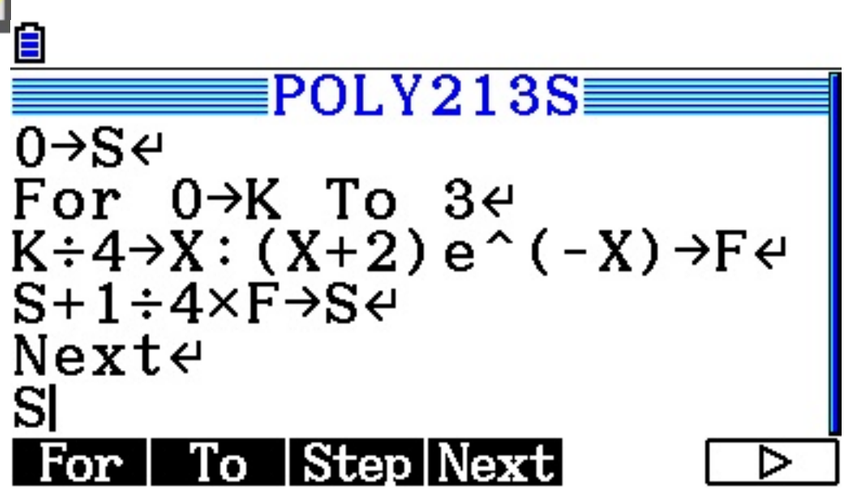
Question 2)a)
On nous propose donc un algorithme calculant une valeur approchée d'une intégrale par la méthode des rectangles.
On nous demande donc ce qu'il affiche, et il nous suffit de le programmer sur notre calculatrice graphique pour avoir la réponse.

Toutefois ici, l'algorithme utilise une fonction f dans une affectation.
On peut faire l'affectation en question telle quelle sur une calculatrice utilisant un langage de programmation fonctionnel comme la TI-Nspire.

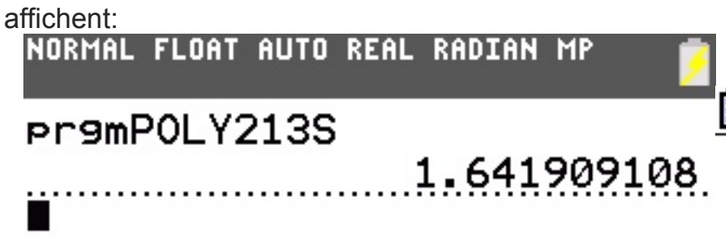
Les autres calculatrices comme les TI-82 à TI-84 et Casio Graph/Prizm utilisent un langage de programmation procédural, ce qui va nécessiter quelques petites adaptations.

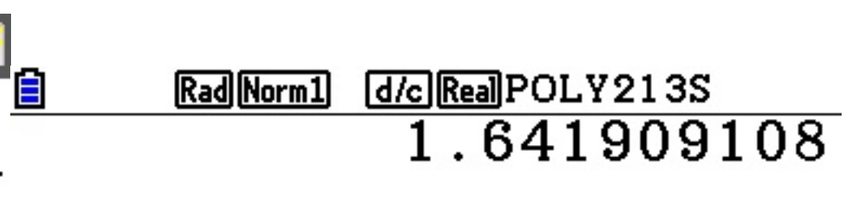
Voici donc des traductions possibles de l'algorithme pour toute calculatrice TI-76/82/83/84 et Casio Graph/Prizm:





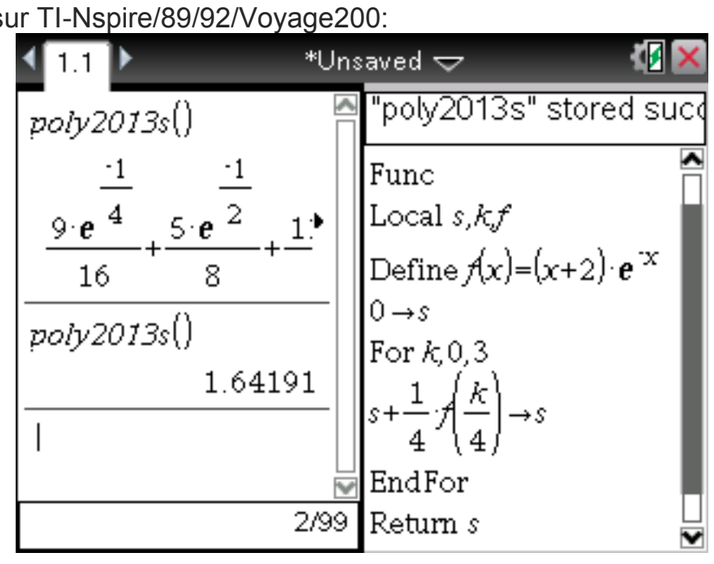
Et voici maintenant ce qu'ils affichent:





La réponse est donc 1,642 à 10^{-3} près.

La même chose est donc réalisable beaucoup plus aisément sur TI-Nspire/89/92/Voyage200:



Si nous sommes sur une TI-Nspire CAS ou une TI-89/92/Voyage200, par défaut la calculatrice utilise son moteur de calcul formel et donne un résultat exact.

Mais il suffit par exemple sur TI-Nspire CAS de lancer le programme en forçant l'utilisation du moteur de calcul numérique approché, en validant avec ctrl ≈ enter

Question 2)b)
L'algorithme précédent utilise donc 4 rectangles pour déterminer son approximation.
On aimerait ici une modification utilisant un nombre N de rectangles.

Pack et mise en page par Laurae, rédigé par critor sur le forum, administrateurs de TI-Planet.org

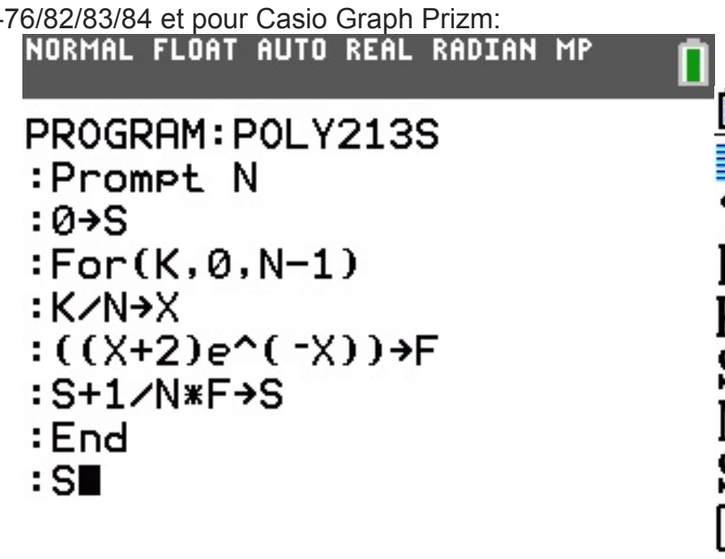
Il va donc suffire de saisir le nombre de rectangles voulus (instruction d'entrée) en début d'algorithme.
Puisque pour N=4 les bornes de la boucle 'pour' étaient 0 et 3, pour N rectangles les bornes sont donc 0 et N-1.

Voici l'algorithme modifié en conséquence:

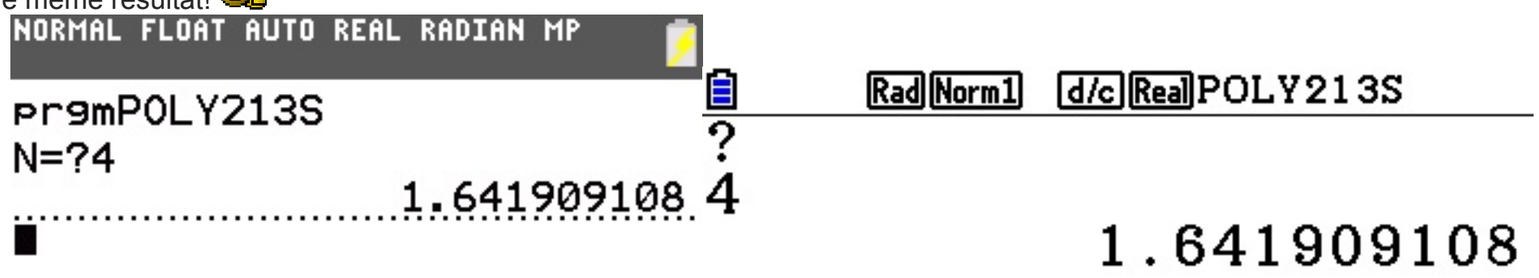
```
Variables:
   k est un nombre entier
   S est un nombre réel
Entrées:
   Saisir N
Initialisation:
   Affecter à S la valeur 0
Traitement:
   Pour k variant de 0 à N-1
      Affecter à S la valeur S+1/N*f(k/N)
   FinPour
Sortie:
   Afficher S
```

On peut vérifier astucieusement la justesse de cet algorithme en modifiant en conséquence notre programme sur calculatrice, et en vérifiant qu'il continue de retourner la même valeur pour N=4:

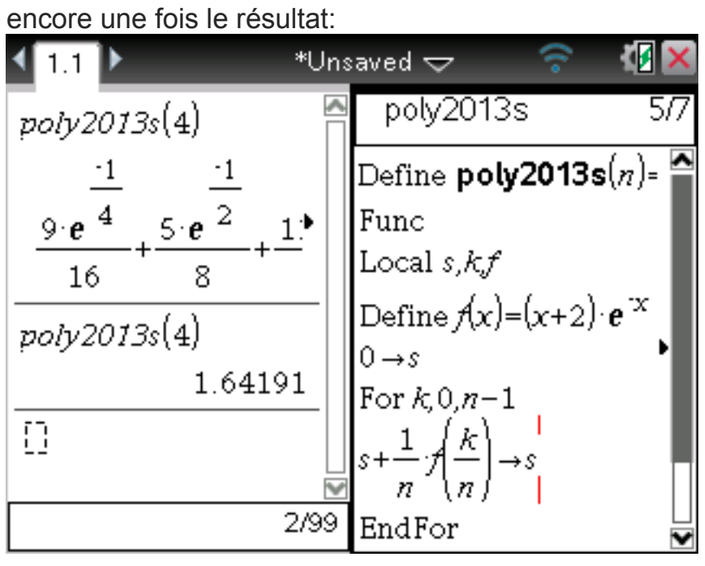
Voici la modification pour TI-76/82/83/84 et pour Casio Graph Prizm:



Effectivement, c'est toujours le même résultat!



Et voici maintenant la modification pour TI-Nspire, qui confirme encore une fois le résultat:



A bientôt

Pack et mise en page par Laurae, rédigé par critor sur le forum, administrateurs de TI-Planet.org

Correction 33 : Correction algorithme sujet de Maths BAC ES/L 2013 Polynésie

de **critor** » 08 Juin 2013 16:52

Rebonjour!

Intéressons-nous maintenant à l'algorithme qui est tombé hier en exercice 3 pour les candidats du BAC ES/L 2013 en Polynésie française.

Jusqu'à présent, chaque sujet du BAC S/ES/L 2013 est tombé avec un algorithme et il est donc très important de bien comprendre ce qui suit, pour pouvoir le réappliquer dans le contexte qui vous sera imposé la semaine du BAC!

EXERCICE 3 **(5 points)**

Commun à tous les candidats

La production des perles de culture de Tahiti est une activité économique importante pour la Polynésie Française.

Les montants réalisés à l'exportation des produits perliers de 2008 à 2011 sont donnés dans le tableau suivant, en milliers d'euros :

Années	2008	2009	2010	2011
Valeurs brutes des produits perliers (en milliers d'euros)	81 295	66 052	64 690	63 182

Source : ISPF (Institut de Statistiques de Polynésie Française

1. Montrer que le taux d'évolution annuel moyen des montants à l'exportation des produits perliers de Polynésie entre 2008 et 2011 est – 8,06% arrondi au centième.

   On admet pour la suite de l'exercice, que la production continuera à baisser de 8 % par an à partir de 2011.

2. On considère l'algorithme suivant :

Entrée	Saisir un nombre positif P
Traitement :	Affecter la valeur 0 à la variable N {initialisation} Affecter la valeur 63 182 à U {initialisation} Tant que U > P Affecter la valeur N + 1 à N Affecter la valeur 0,92 ×U à U Fin de Tant que Affecter la valeur N + 2011 à N
Sortie	Afficher N

Si on saisit P = 50 000 en entrée, qu'obtient-on en sortie par cet algorithme ? Interpréter ce résultat dans le contexte de la production de perles.

Il s'agit donc encore une fois d'un algorithme tombant dans le contexte de suites.

Question 2:

On nous demande donc ce que répond l'algorithme lorsque l'on saisit P=50000.
Il nous suffit pour cela de le programmer sur notre calculatrice graphique.

Voici deux programmes traduisant cet algorithme sur calculatrices TI-76/82/83/84 et Casio Graph/Prizm:

```
NORMAL FLOAT AUTO REAL RADIAN MP
PROGRAM:POL213ES
:Prompt P
:0→N
:63182→U
:While U>P
:N+1→N
:0.92U→U
:End
:N+2011→N
:N■
```

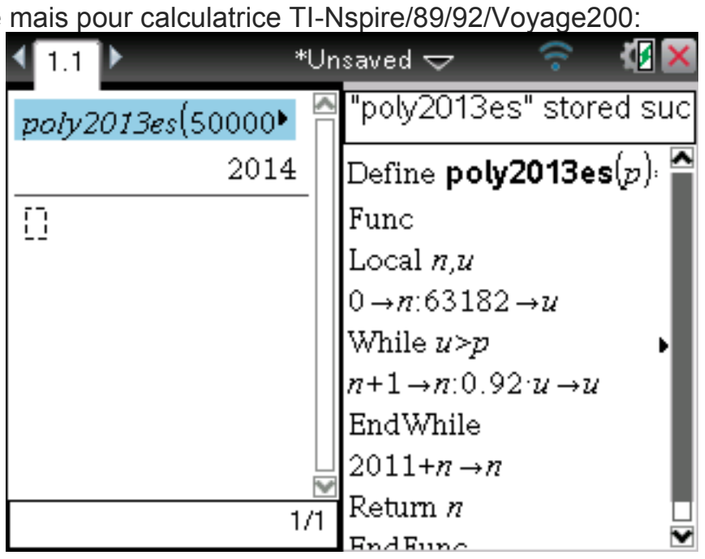
```
POL213ES
?→P:0→N:63182→U↵
While U>P↵
N+1→N:0.92U→U↵
WhileEnd↵
N+2011→N↵
N
TOP BOTTOM SEARCH MENU A⇔a CHAR
```

Dans les deux cas la réponse affichée est 2014:

```
NORMAL FLOAT AUTO REAL RADIAN MP
prgmPOL213ES
P=?50000
                                  2014
■
```

```
Rad Norm1 d/c Real POL213ES
?
50000
                                  2014
```

Voici maintenant un programme traduisant le même algorithme mais pour calculatrice TI-Nspire/89/92/Voyage200:



On nous demande donc d'interpréter.

Le 63182 affecté à U en initialisation est la valeur brute des produits perliers en 2011 en milliers d'euros. On peut donc supposer que la variable U contient la valeur brute des produits perliers.

L'algorithme répond le contenu de la variable N, qui est initialisée à 0 et incrémentée de 2011 juste avant la fin de l'algorithme.
On peut donc supposer que l'algorithme répond une année, et pendant presque tout l'algorithme la variable N contient le rang de l'année en question, l'année 2011 ayant pour rang 0.

L'algorithme s'articule autour d'une boucle 'tant que', dont la condition de poursuite est "U>P", où P est la valeur saisie c'est-à-dire 50000. La condition se réécrit donc "U>50000" dans ce cas.
L'algorithme se termine donc en sortant de la boucle 'tant que' sur la condition contraire: "U≤50000".

L'algorithme estime donc la première année où la valeur brute des produits perliers de Polynésie deviendra inférieure ou égale à 50000 milliers d'euros.

Pack et mise en page par Laurae, rédigé par critor sur le forum, administrateurs de TI-Planet.org

Correction 34 : Correction Algorithme BAC STI2D/STL(SPCL) 2013 (Polynésie)

de **critor** » 11 Juin 2013 19:27

Comme c'est désormais le cas à chaque sujet pour les séries générales, ça n'a pas manqué: le sujet de Maths tombé la semaine dernière pour les candidats au BAC Technologique 2013 Filières STI2D et STL spécialité Sciences Physique et Chimique de Laboratoire contenait un algorithme!

Et mon intuition est que ce ne sera pas le dernier

L'algorithme est tombé en exercice 2, comme d'habitude un exercice de suites.
Le voici:

EXERCICE 2 **(5 points)**

On considère la suite numérique (u_n) définie par :

$u_0 = 8$ et, pour tout entier naturel n, $u_{n+1} = 0{,}4\,u_n + 3$.

1. Calculer u_1 et u_2.
   On utilise un tableur pour calculer les premiers termes de cette suite.
   Une copie d'écran sur laquelle les termes u_1 et u_2 ont été effacés est donnée en **annexe 1 (page 7)**.

2. Quelle formule a-t-on pu saisir dans la cellule B3 de la feuille de calcul afin d'obtenir les premiers termes de cette suite par recopie vers le bas ?
3. En utilisant cette copie d'écran, que peut-on conjecturer sur la limite de la suite (u_n) ?
4. On considère l'algorithme suivant :

Les variables sont l'entier naturel N et le réel U.	
Initialisation :	Affecter à N la valeur 0
	Affecter à U la valeur 8
Traitement :	TANT QUE U − 5 > 0,01
	Affecter à N la valeur N + 1
	Affecter à U la valeur 0,4U + 3
	Fin TANT QUE
Sortie :	Afficher N

Par rapport à la suite (u_n), quelle est la signification de l'entier N affiché ?

Question 3:
L'algorithme utilise donc:
- une variable N initialisée à 0 (Affecter à N la valeur 0) puis incrémentée de 1 en 1 (Affecter à N la valeur N+1)
- une variable U initialisée à 8 (Affecter à U la valeur 8) puis modifiée par récurrence (Affecter à U la valeur 0,4U+3)

Tout ceci doit nous faire penser à la suite définie en début d'énoncé:
$u_0=8$
$u_{n+1}=0{,}4u_n+3$

Cet algorithme calcule et utilise donc les termes de la suite:
- N est le rang
- U contient alors la valeur du terme u_N

Mais que fait donc cet algorithme?
Il est constitué essentiellement d'une boucle 'tant que' dont la condition de poursuite est U-5>0,01.

Lorsque l'algorithme se termine, c'est donc qu'il y a eu réalisation de la proposition contraire: U-5≤0,01.

L'algorithme retourne donc le rang n du premier terme de la suite u vérifiant u_n-5≤0,01, c'est-à-dire u_n≤5,01.

On a pu conjecturer en question 3 que la suite u était strictement décroissante et convergeait vers une limite de 5.

Dans le contexte de cette conjecture, on est donc sûr que l'algorithme s'arrêtera, et renverra le rang d'un terme vérifiant $5<u_n\leq 5{,}01$.

Bonne réussite!

Pack et mise en page par Laurae, rédigé par critor sur le forum, administrateurs de TI-Planet.org

Correction 35 : Correction algo BAC STL(Biotechnologies) 2013 (Polynésie)

de **critor** » 12 Juin 2013 13:46

Après la correction de l'algorithme tombé en Polynésie française au sujets de Maths commun aux nouvelles séries technologiques du BAC 2013 STI2D et STL spécialité Sciences Physiques et Chimiques de Laboratoire, voici aujourd'hui la correction de l'algorithme tombé en STL spécialité Biotechnologies.

L'algorithme est tombé en exercice 1, dans le contexte d'un QCM sans pénalité ni justification, que voici:

EXERCICE 2 (3 points)

Cet exercice est un questionnaire à choix multiples. Pour chacune des questions posées, une seule des quatre réponses est exacte.

Recopier le numéro de chaque question et préciser la réponse choisie. Aucune justification n'est demandée.

Une réponse exacte rapporte 1 point ; une réponse fausse ou l'absence de réponse ne rapporte ni n'enlève aucun point.

1. On considère l'algorithme suivant (N désigne un entier naturel)

Entrée : Saisir la valeur de N
Initialisation : Affecter à i la valeur 0
Affecter à A la valeur 25
Traitement : Tant que $i < N$
Affecter à i la valeur de $i + 1$
Affecter à A la valeur de $1{,}05 \times A - 0{,}1$
Fin Tant que
Sortie : Afficher A

Pour $N = 4$, l'arrondi à 10^{-2} du nombre affiché est :

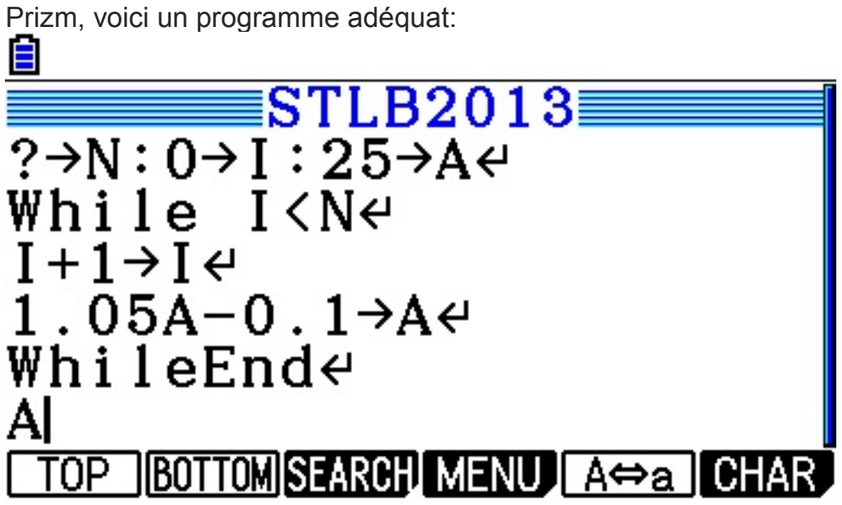
a) 26,15 b) 28,63 c) 29,96 d) 30,82.

On nous demande donc juste de cocher ce qu'affiche l'algorithme à 10^{-2} près.

Et bien il nous suffit tout simplement de la traduire en un programme pour notre calculatrice graphique et de lire la réponse!

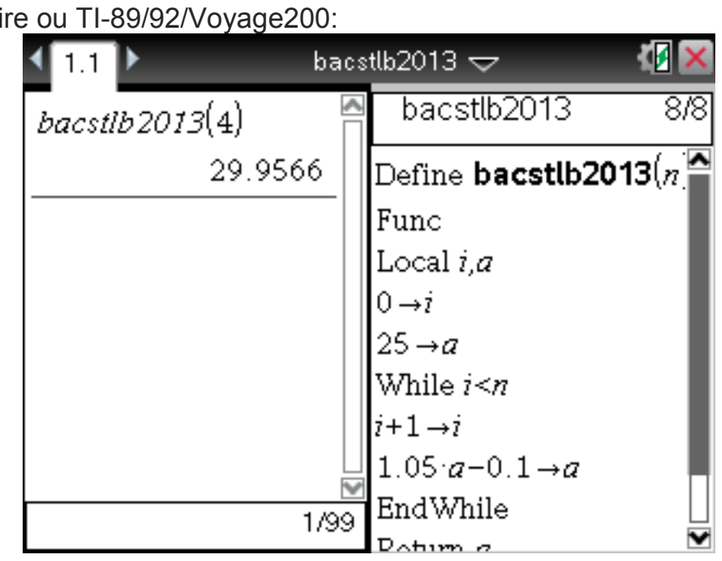
Voici ce que l'on peut faire avec une calculatrice TI-76 ou TI-82 à TI-84:

Ou si jamais vous êtes muni d'une Casio Graph ou Casio Prizm, voici un programme adéquat:

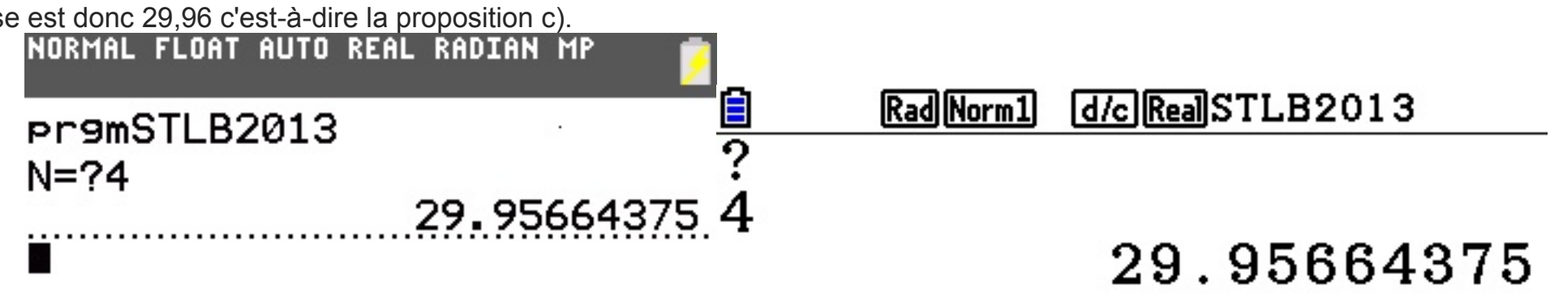


Pack et mise en page par Laurae, rédigé par critor sur le forum, administrateurs de TI-Planet.org

Voici enfin une adaptation possible de l'algorithme sur TI-Nspire ou TI-89/92/Voyage200:



Dans tous les cas, la réponse est donc 29,96 c'est-à-dire la proposition c).



Bonne réussite!

Pack et mise en page par Laurae, rédigé par critor sur le forum, administrateurs de TI-Planet.org

Correction 36 : Correction algo Spécialité du BAC S 2013 (Centres Etrangers)

de **critor** » 12 Juin 2013 15:58

L'exercice de spécialité du sujet de Maths du BAC S 2013 tombé ce matin pour les candidats des lycées français des Centres Etrangers (Europe/Afrique) comportaire donc un algorithme.

Le voici:

Exercice 4 (5 points)

Candidats ayant choisi la spécialité mathématique

Une espèce d'oiseaux ne vit que sur deux îles A et B d'un archipel.
Au début de l'année 2013, 20 millions d'oiseaux de cette espèce sont présents sur l'île A et 10 millions sur l'île B.
Des observations sur plusieurs années ont permis aux ornithologues d'estimer que, compte tenu des naissances, décès, et migrations entre les deux îles, on retrouve au début de chaque année les proportions suivantes :

- sur l'île A : 80 % du nombre d'oiseaux présents sur l'île A au début de l'année précédente et 30 % du nombre d'oiseaux présents sur l'île B au début de l'année précédente ;
- sur l'île B : 20 % du nombre d'oiseaux présents sur l'île A au début de l'année précédente et 70 % du nombre d'oiseaux présents sur l'île B au début de l'année précédente.

Pour tout entier naturel n, on note a_n (respectivement b_n) le nombre d'oiseaux (en millions) présents sur l'île A (respectivement B) au début de l'année $(2013+n)$.

Partie A – Algorithmique et conjectures

On donne ci-contre un algorithme qui doit afficher le nombre d'oiseaux vivant sur chacune des deux îles, pour chaque année comprise entre 2013 et une année choisie par l'utilisateur.

```
Début de l'algorithme
Lire n
Affecter à a la valeur 20
Affecter à b la valeur 10
Affecter à i la valeur 2013
Afficher i
Afficher a
Afficher b
Tant que i < n faire
    Affecter à c la valeur (0,8a + 0,3b)
    Affecter à b la valeur (0,2a + 0,7b)
    Affecter à a la valeur c
Fin du Tant que
Fin de l'algorithme
```

1. Cet algorithme comporte des oublis dans le traitement. Repérer ces oublis et les corriger.

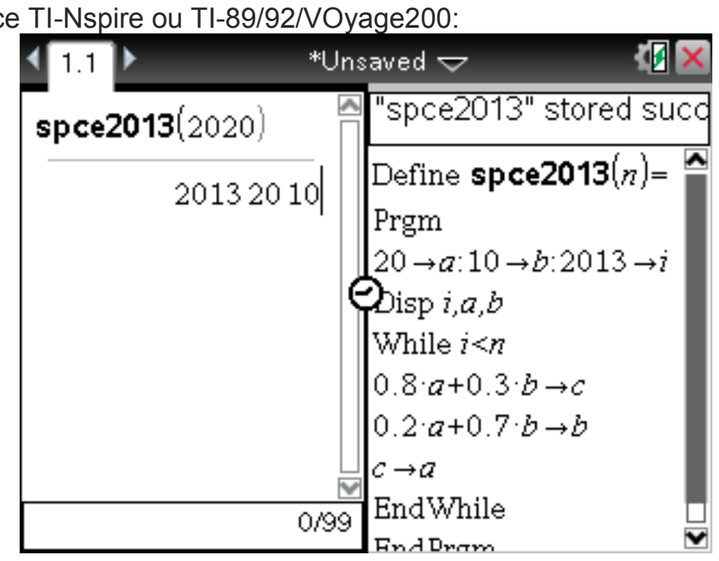
2. On donne ci-dessous une copie d'écran des résultats obtenus après avoir corrigé l'algorithme précédent dans un logiciel d'algorithmique, l'utilisateur ayant choisi l'année 2020.

```
***Algorithme lancé***
En l'année 2013, a prend la valeur 20 et b prend la valeur 10
En l'année 2014, a prend la valeur 19 et b prend la valeur 11
En l'année 2015, a prend la valeur 18.5 et b prend la valeur 11.5
En l'année 2016, a prend la valeur 18.25 et b prend la valeur 11.75
En l'année 2017, a prend la valeur 18.125 et b prend la valeur 11.875
En l'année 2018, a prend la valeur 18.0625 et b prend la valeur 11.9375
En l'année 2019, a prend la valeur 18.03125 et b prend la valeur 11.96875
En l'année 2020, a prend la valeur 18.015625 et b prend la valeur 11.984375
***Algorithme terminé***
```

Question A)1)
Cet algorithme est censé lister le nombre d'oiseaux vivant sur deux îles chaque année.
Pour une fois on nous donne donc un algorithme faux et on nous demande de le corriger.

Pack et mise en page par Laurae, rédigé par critor sur le forum, administrateurs de TI-Planet.org

Voyons déjà ce qui ne va pas en le tapant sur notre calculatrice TI-Nspire ou TI-89/92/VOyage200:



En fait le programme traduisant l'algorithme comporte plusieurs problèmes:
- P1: il n'affiche que les populations d'oiseaux initiales en 2013 et ne les liste donc pas pour les années suivantes
- P2: il se s'arrête jamais

Les mêmes problèmes sont constatables en traduisant l'algorithme pour un autre modèle comme une TI-76, TI-82 à TI-84 ou encore Casio Graph/Prizm:

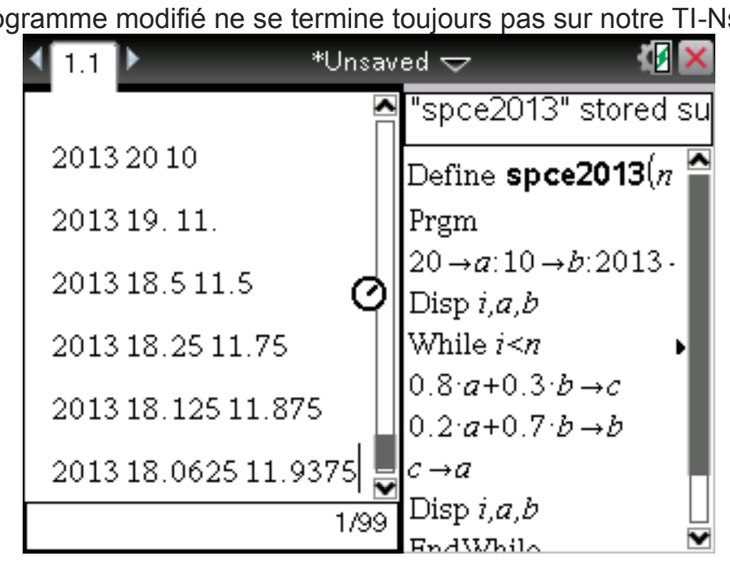
Le problème P1 est en effet normal.
Il n'y a qu'un seul groupe d'instructions de sortie pour i, a et b, et il se situe avant la boucle.
Il n'est donc exécuté qu'une seule fois et ne produit l'affichage que d'un seul triplet.

Il nous faut rajouter un groupe d'instructions de sortie similaires dans la boucle.
Je propose donc la modification suivante:

CODE: TOUT SÉLECTIONNER

```
Lire n
Affecter à a la valeur 20
Affecter à b la valeur 10
Affecter à i la valeur 2013
Afficher i
Afficher a
Afficher b
Tant que i<n faire
   Affecter à c la valeur 0,8a+0,3b
   Affecter à b la valeur 0,2a+0,7b
   Affecter à a la valeur c
   Afficher i
   Afficher a
   Afficher b
Fin du Tant que
```

Cette fois-ci nous avons bien plusieurs affichages, mais le programme modifié ne se termine toujours pas sur notre TI-Nspire:



Même constat sur les TI-76/82-84 et Casio Graph/Prizm:

Pack et mise en page par Laurae, rédigé par critor sur le forum, administrateurs de TI-Planet.org

Reste donc maintenant à régler le problème P2 de cet algorithme qui ne se termine pas.
Etant construit autour d'une boucle 'Tant que', cela veut dire que la boucle est infinie, c'est-à-dire que la condition de poursuite i<n reste éternellement vraie.

Et en effet, il n'y a au sein de cette boucle aucune affectation pouvant modifier les valeurs de i ou n.

i étant initialisé à 2013, il s'agit de l'année affichée à chaque fois.
Et en effet, on constate ci-dessus sur TI-Nspire que le programme n'affiche que des 2013, bien que les populations d'oiseaux varient.

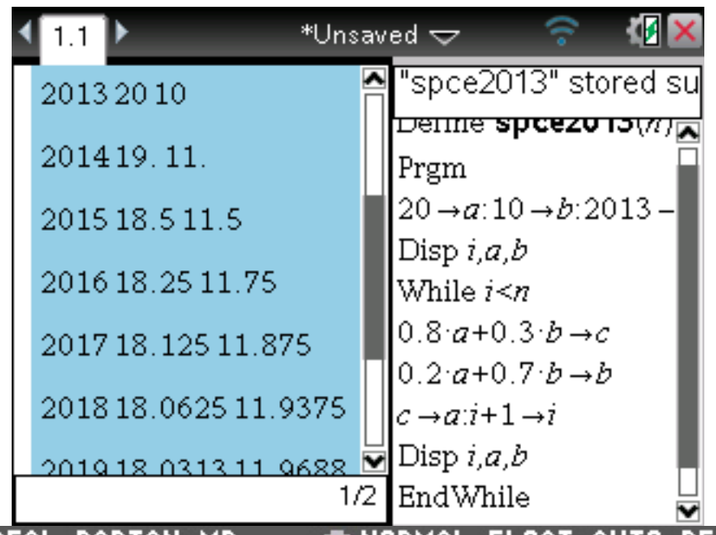
Il manque donc tout simplement à incrémenter la variable i avant l'affichage de la boucle, afin que l'affichage des populations puisse être correct et que la boucle puisse se terminer lorsque i aura atteint la valeur n saisie (2020).

Je propose:

```
Lire n
Affecter à a la valeur 20
Affecter à b la valeur 10
Affecter à i la valeur 2013
Afficher i
Afficher a
Afficher b
Tant que i<n faire
   Affecter à c la valeur 0,8a+0,3b
   Affecter à b la valeur 0,2a+0,7b
   Affecter à a la valeur c
   Affecter à i la valeur i+1
   Afficher i
   Afficher a
   Afficher b
Fin du Tant que
```

Et cette fois-ci, ça marche sur tous les modèles et on peut constater la production d'un affichage conforme à celui qui est gentiment donné par la question 2!

Pack et mise en page par Laurae, rédigé par critor sur le forum, administrateurs de TI-Planet.org



Question A)2)

Au vu des valeurs listées, on conjecture que:

- la suite a_n est strictement décroissante
- la suite b_n est strictement croissante
- la suite a_n converge vers 18

la suite b_n converge vers 12

Pack et mise en page par Laurae, rédigé par critor sur le forum, administrateurs de TI-Planet.org

Correction 37 : Correction algorithme Maths BAC ES/L 2013 Centres Etrangers

de **critor** » 13 Juin 2013 19:21

Regardons un petit peu aujourd'hui l'algorithme qui vient de tomber hier dans l'épreuve de Maths commune aux séries ES Obligatoire et L Spécialité, pour les candidats passant les BAC ES et L 2013 dans les lycées français des Centres Etrangers (Europe et Afrique).

Encore une fois, comme dans une écrasante majorité des cas au BAC, l'algorithme tombe dans le contexte d'un exercice sur les suites, mais petite originalité cette fois-ci dans un QCM sans justification ni pénalité en questions 4 et 5:

EXERCICE 1 (5 points)

Commun à tous les candidats

Les services de la mairie d'une ville ont étudié l'évolution de la population de cette ville. Chaque année, 12,5 % de la population quitte la ville et 1 200 personnes s'y installent.
En 2012, la ville comptait 40 000 habitants.

On note U_n le nombre d'habitants de la ville en l'année $2012 + n$.
On a donc $U_0 = 40\,000$.
On admet que la suite (U_n) est définie pour tout entier naturel n par $U_{n+1} = 0{,}875 \times U_n + 1\,200$.

On considère la suite (V_n) définie pour tout entier naturel n par $V_n = U_n - 9\,600$.

Les questions numérotées de 1 à 5 de cet exercice forment un questionnaire à choix multiples (QCM). Pour chacune des questions, quatre affirmations sont proposées ; une seule réponse est exacte. Une réponse exacte rapporte 1 point, une réponse fausse ou l'absence de réponse ne rapporte ni n'enlève aucun point. Pour chaque question, le candidat notera sur sa copie le numéro de la question suivi de la proposition qui lui semble correcte. Aucune justification n'est demandée.

1) La valeur de U_1 est :

a) 6 200 **b)** 35 000 **c)** 36 200 **d)** 46 200

2) La suite (V_n) est :

a) géométrique de raison – 12,5 %
b) géométrique de raison 0,875
c) géométrique de raison – 0,875
d) arithmétique de raison – 9 600

3) La suite (U_n) a pour limite :

a) $+\infty$ **b)** 0 **c)** 1 200 **d)** 9 600

4) On donne l'algorithme suivant :

```
Variables :
        U, N
Initialisation :
        U prend la valeur 40 000
        N prend la valeur 0
Traitement :
        Tant que U > 10 000
                N prend la valeur N + 1
                U prend la valeur 0,875 × U + 1 200
        Fin du Tant que
Sortie :
        Afficher N
```

Cet algorithme permet d'obtenir :

a) la valeur de $U_{40\,000}$
b) toutes les valeurs de U_0 à U_N
c) le plus petit rang n pour lequel on a $U_n \leq 10\,000$
d) le nombre de termes inférieurs à 1 200

5) La valeur affichée est :

a) 33 **b)** 34 **c)** 9 600 **d)** 9 970,8

Pack et mise en page par Laurae, rédigé par critor sur le forum, administrateurs de TI-Planet.org

Question 4)
Il s'agit donc d'interpréter ce que fait l'algorithme, en choisissant la bonne réponse:
- **a)** la valeur de U_{40000}
- **b)** toutes les valeurs de U_0 à U_N
- **c)** le plus petit rang n pour lequel on a $U_n \leq 10000$
- **d)** le nombre de termes inférieurs à 1200

L'algorithme de l'énoncé comporte l'affectation récurrente "U prend la valeur 0,875U+1200" correspondant à la relation de récurrence donnée en introduction pour la suite: $U_{n+1}=0{,}875U_n+1200$.
De plus, on y trouve l'initialisation "U prend la valeur 40000" qui correspond au terme initial $U_0=40000$.
Cet algorithme travaille donc sur la suite (U_n) de l'énoncé.

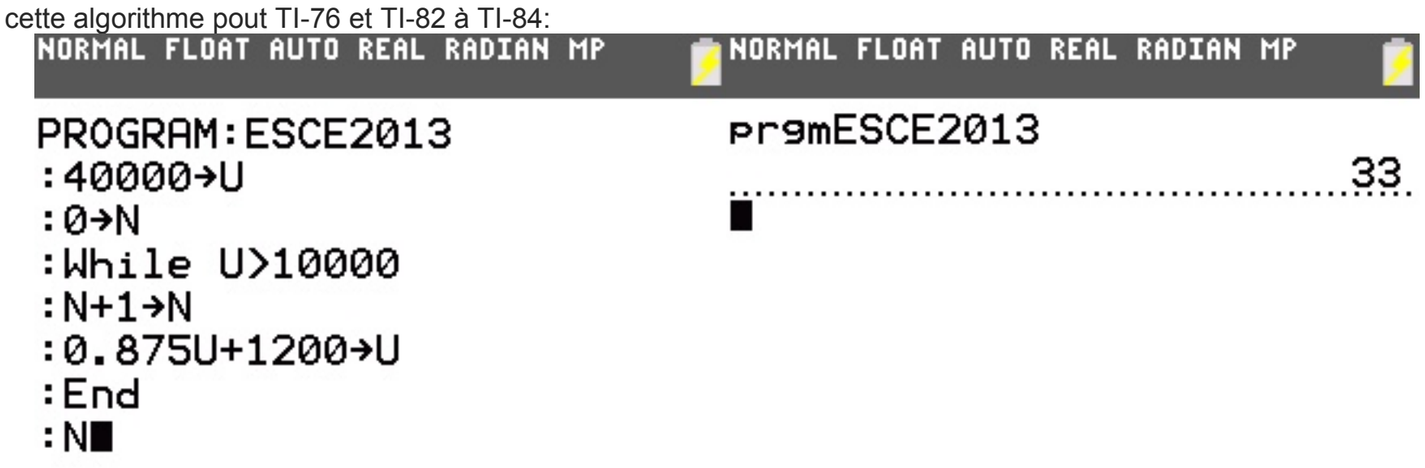
Donc, que fait-il avec?
L'algorithme est principalement constitué d'une boucle 'tant que'.
Sa condition de poursuite est U>10000, ce qui nous donne une condition d'arrêt terminant l'algorithme sur son contraire, U≤10000.

L'algorithme recherche donc le rang du premier terme de la suite (U_n) inférieur ou égal à 10000.
Réponse **c)**.

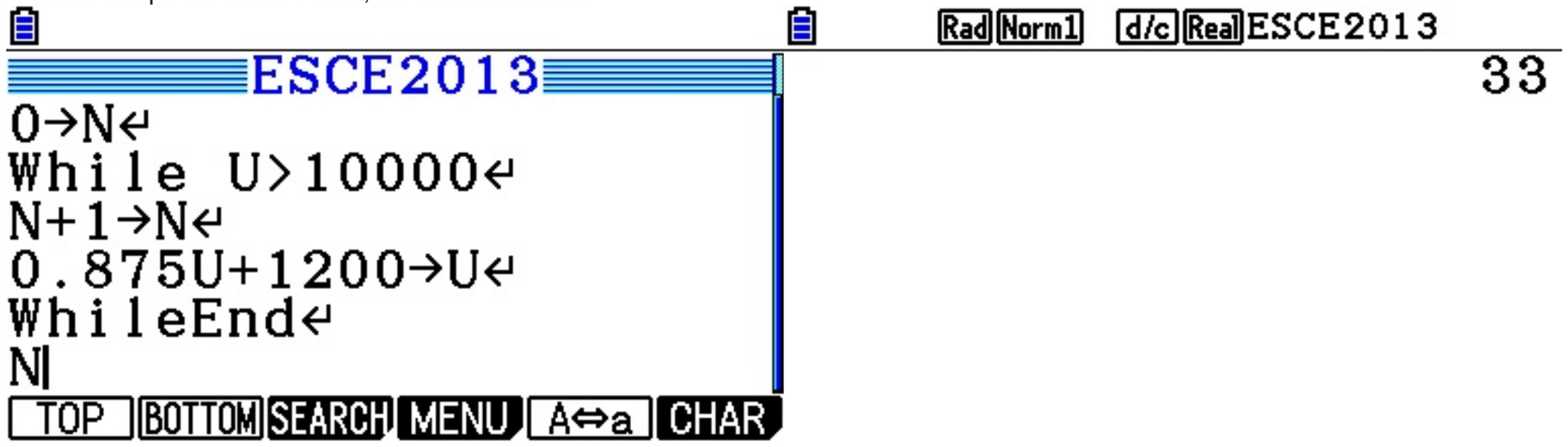
Question 5)
On nous demande maintenant ce qu'affiche cet algorithme.

On peut tout simplement le programmer sur notre calculatrice et recopier la réponse.

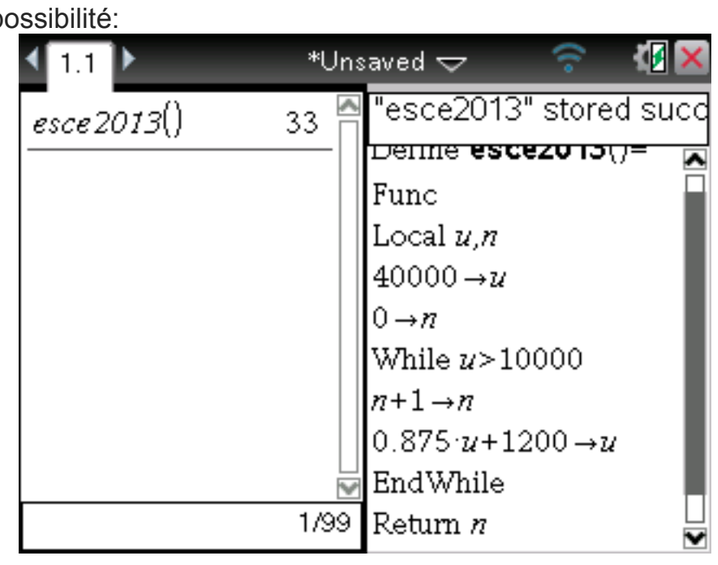
Voici une adaptation possible de cette algorithme pout TI-76 et TI-82 à TI-84:



Si vous êtes équipé d'une Casio Graph ou Casio Prizm, en voici une autre:



Enfin, pour une TI-Nspire ou TI-89/92/Voyage 200, voici une possibilité:



Dans tous les cas, la valeur affichée est donc 33 et la bonne réponse **a)**.

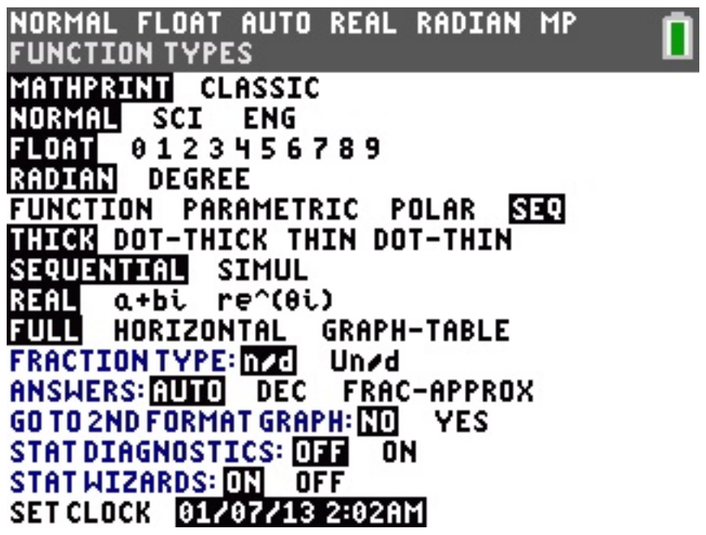
Mais si jamais vous n'étiez, malgré nos efforts, toujours pas à l'aise avec l'implémentation d'algorithmes sur votre calculatrice, il était parfaitement possible de résoudre autrement la question 5) à condition d'avoir compris la question 4)...
Il nous suffit en effet de demander à la calculatrice un tableau de valeurs de la suite, et de rechercher le rang du premier terme de valeur inférieure ou égale à 10000.

Prenons pour commencer une TI-82 à TI-84.

Correction algorithmes

Pack et mise en page par Laurae, rédigé par critor sur le forum, administrateurs de TI-Planet.org

Passons donc en mode 'suite' ou 'sequence' en anglais -

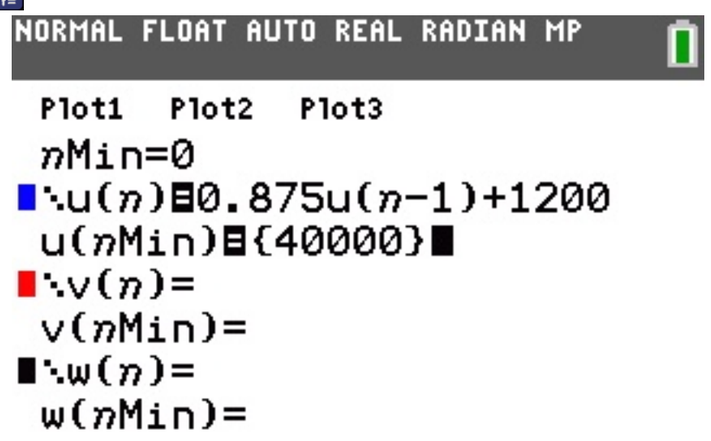


La suite vous est donc définie par une relation de récurrence $U_{n+1}=0,875U_n+1200$.

Mais la calculatrice ne vous permet pas de définir le terme de rang n+1 mais uniquement le terme de rang n.

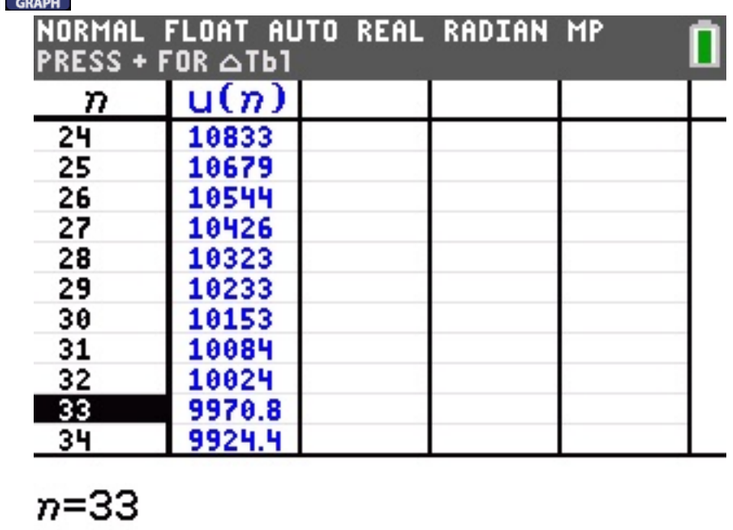
Vous devez donc commencer par réécrire cette relation un rang en-dessous, c'est-à-dire en remplaçant tous les 'n' par des 'n-1'.
Cela nous donne $U_n=0,875U_{n-1}+1200$

Une fois cette relation établie, il vous suffit d'aller la saisir -



Selon l'état de votre calculatrice, il peut alors être nécessaire de modifier les paramètres du tableau de valeurs, à partir de 0 avec un pas de 1 dans le cas d'une suite -

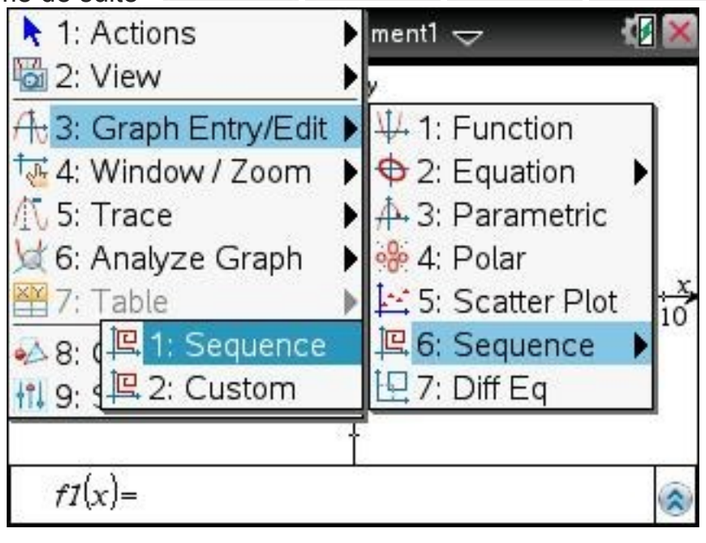
Et vous pouvez enfin demander le tableau de valeurs -



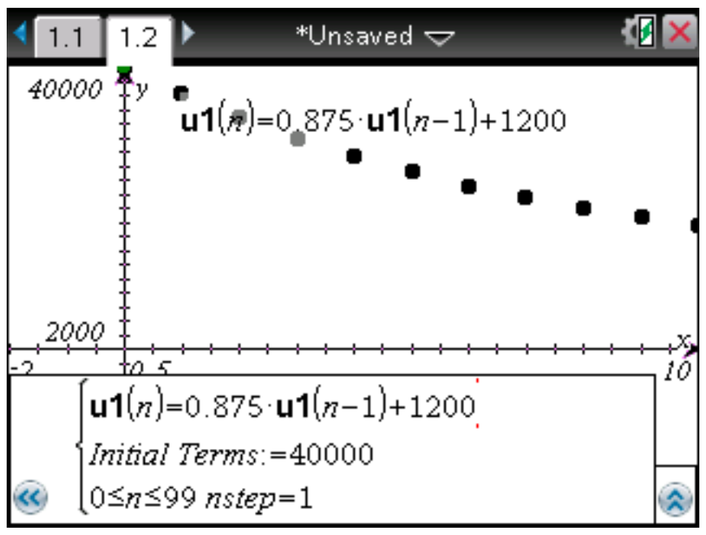
Sur TI-Nspire il vous faut utiliser la même relation.

Pack et mise en page par Laurae, rédigé par critor sur le forum, administrateurs de TI-Planet.org

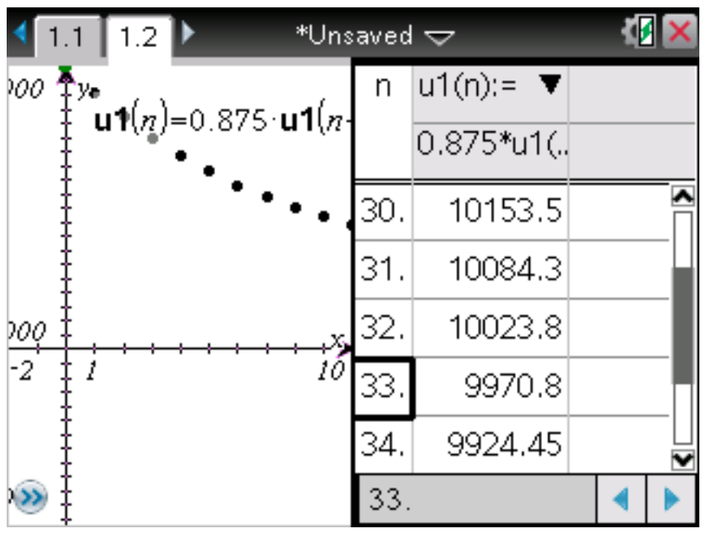
Dans une application graphique, choisissez un entrée sous forme de suite - menu 3 6 1



Saisissez alors la relation trouvée plus haut:

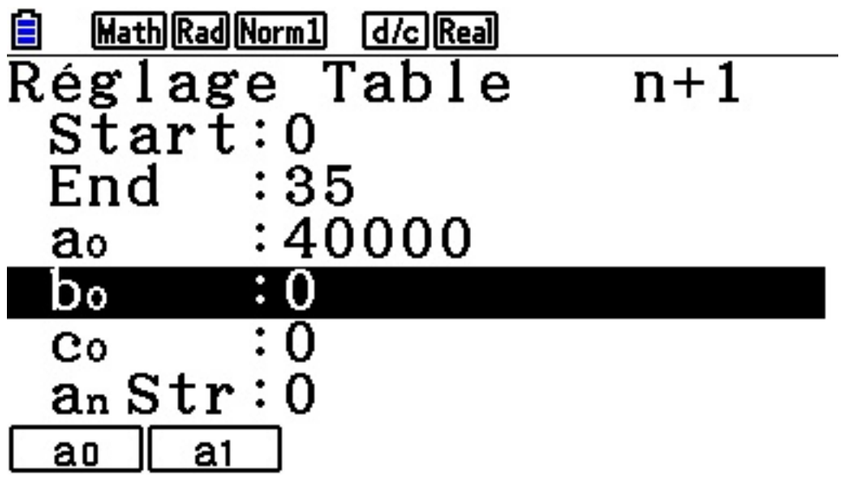


Et demandez le tableau de valeurs - ctrl T

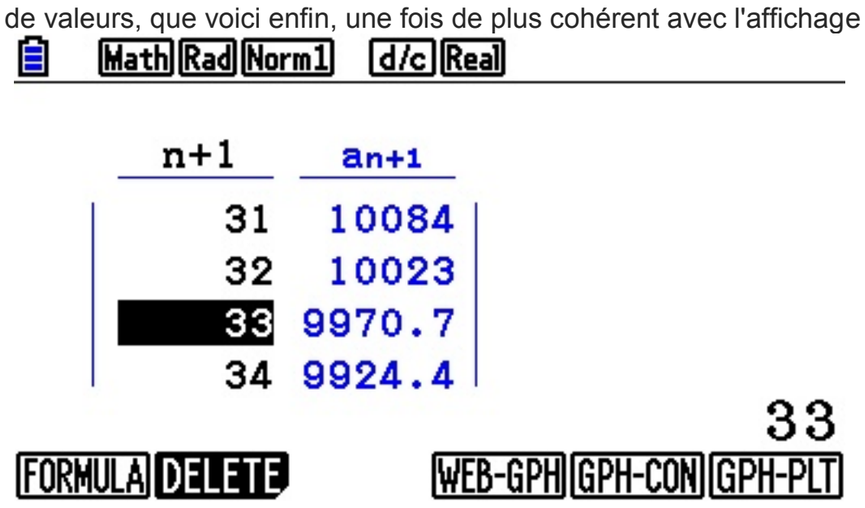


Si vous êtes munis d'une Casio Graph/Prizm, il suffit d'accéder à l'application 'Récurrence' pour y définir notre suite.
Ici, nul besoin de transformer l'expression contrairement aux calculatrices TI - on peut saisir directement celle de l'énoncé!

Mais il ne faut surtout pas oublier de préciser la valeur des termes initiaux, et cela se fait dans un autre écran accessible via le menu [SET]:



On y précise également les paramètres de notre tableau de valeurs, que voici enfin, une fois de plus cohérent avec l'affichage du programme!



La réponse est donc bien 33.

A bientôt!

Pack et mise en page par Laurae, rédigé par critor sur le forum, administrateurs de TI-Planet.org

Correction 38 : Correction algorithme Obligatoire BAC S 2013 (Antilles)

de **critor** » 19 Juin 2013 23:52

Intéressons-nous ce soir à l'algorithme tombé au sujet de Maths Obligatoire du BAC S 2013 en Antilles-Guyane, en exercice 4.

Il est assez particulier et donc intéressant; en effet il n'est pas seulement tombé avec une suite comme d'habitude, mais avec une suite complexe!

EXERCICE 4 (5 points)
candidats n'ayant pas suivi l'enseignement de spécialité

On considère la suite (z_n) à termes complexes définie par : $z_0 = 1 + i$ et, pour tout entier naturel n, par

$$z_{n+1} = \frac{z_n + |z_n|}{3}.$$

Pour tout entier naturel n, on pose : $z_n = a_n + ib_n$, où a_n est la partie réelle de z_n et b_n est la partie imaginaire de z_n.

Le but de cet exercice est d'étudier la convergence des suites (a_n) et (b_n).

Partie A

1. Donner a_0 et b_0.
2. Calculer z_1, puis en déduire que $a_1 = \frac{1+\sqrt{2}}{3}$ et $b_1 = \frac{1}{3}$.
3. On considère l'algorithme suivant :

```
Variables :        A et B des nombres réels
                   K et N des nombres entiers
Initialisation :   Affecter à A la valeur 1
                   Affecter à B la valeur 1
Traitement :
Entrer la valeur de N
Pour K variant de 1 à N
        Affecter à A la valeur (A+√(A²+B²))/3
        Affecter à B la valeur B/3
FinPour
Afficher A
```

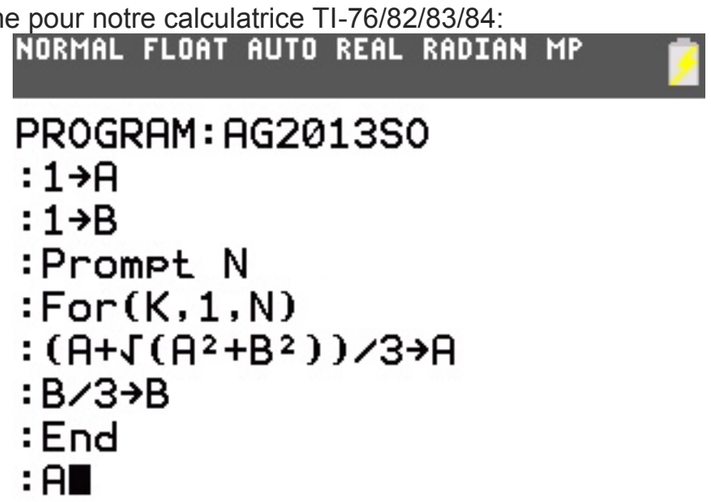
a. On exécute cet algorithme en saisissant N = 2. Recopier et compléter le tableau ci-dessous contenant l'état des variables au cours de l'exécution de l'algorithme (on arrondira les valeurs calculées à 10^{-4} près).

K	A	B
1		
2		

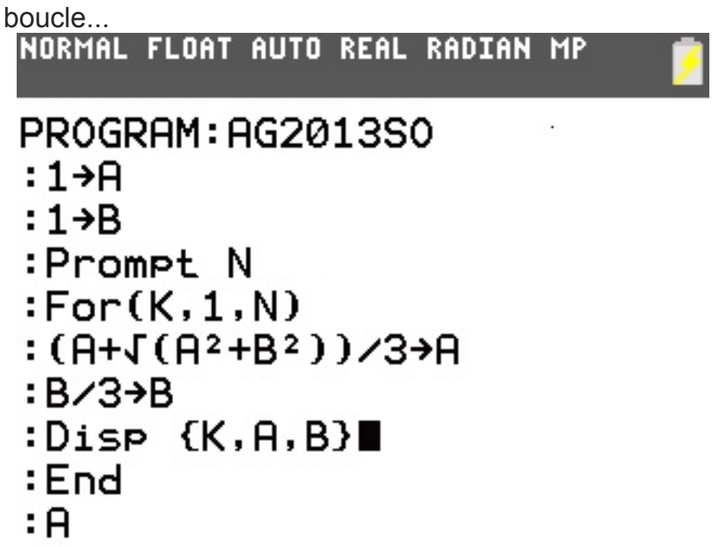
b. Pour un nombre N donné, à quoi correspond la valeur affichée par l'algorithme par rapport à la situation étudiée dans cet exercice ?

Question A)3)a)
On nous demande donc une trace de l'algorithme avec l'état des variables à chaque itération de la boucle (supposons que c'est à chaque fin d'itération de boucle - puisque l'énoncé ne précise pas).

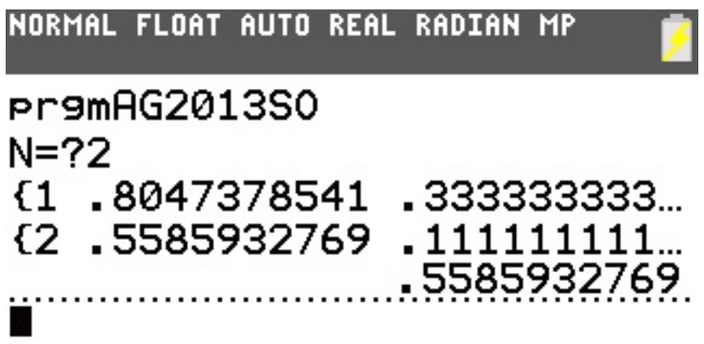
Il nous suffit pour cela de traduire l'algorithme en un programme pour notre calculatrice TI-76/82/83/84:



Rajoutons maintenant une simple instruction de sortie dans la boucle...

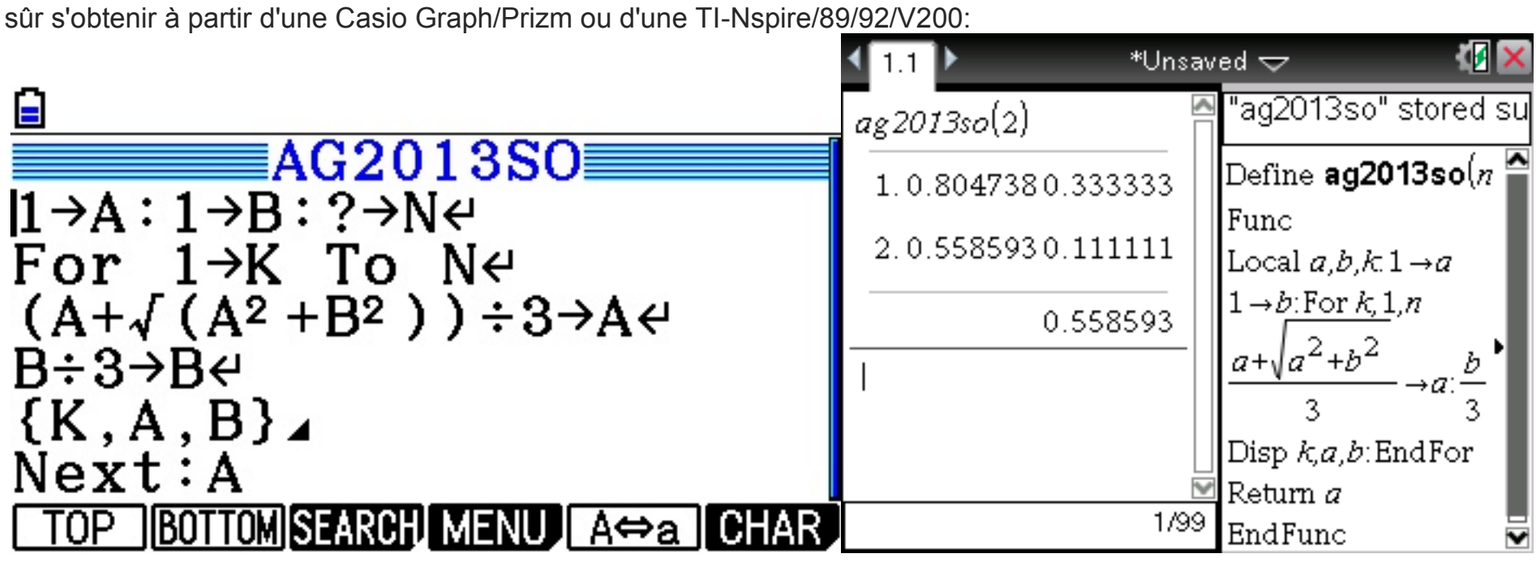


Et il n'y a plus qu'à recopier les résultats!



K	A	B
1	0,8047	0,3333
2	0,5586	0,1111

La même chose pouvait bien sûr s'obtenir à partir d'une Casio Graph/Prizm ou d'une TI-Nspire/89/92/V200:



Question A)3)b)

L'énoncé définit donc une suite complexe $z_{n+1}=(z_n+|z_n|)/3$ avec $z_0=1+i$.

Cette suite complexe peut être modélisée par deux suites réelles a et b prenant respectivement les valeurs des parties réelles et imaginaires:
$a_{n+1}=(a_n+\sqrt{(a_n^2+b_n^2)})/3$ avec $a_0=1$
$b_{n+1}=b_n/3$ avec $b_0=1$

Ce sont ces suites qu'implémente l'algorithme, ces relations se retrouvant exactement dans les initialisations et affectations récurrentes sur A et B.
On pouvait aussi deviner en remarquant $b_1=1/3$ dans la trace précédente.

Comme l'algorithme se termine par un affichage de A, son but est d'afficher a_n, ou encore la partie réelle de z_n.

Pack et mise en page par Laurae, rédigé par critor sur le forum, administrateurs de TI-Planet.org

Correction 39 : Correction algorithme BAC S 2013 (France/Réunion)

de **critor** » 20 Juin 2013 12:10

L'algorithme tombé ce matin en France et Réunion à l'épreuve de Maths du BAC S 2013 ne portait donc pas sur les suites mais sur les fonctions.

Le voici:

4. On donne l'algorithme ci-dessous.

Variables :	a, b et m sont des nombres réels.
Initialisation :	Affecter à a la valeur 0.
	Affecter à b la valeur 1.
Traitement :	Tant que $b-a>0,1$
	Affecter à m la valeur $\frac{1}{2}(a+b)$.
	Si $f(m)<1$ alors Affecter à a la valeur m.
	Sinon Affecter à b la valeur m.
	Fin de Si.
	Fin de Tant que.
Sortie :	Afficher a.
	Afficher b.

a. Faire tourner cet algorithme en complétant le tableau ci-dessous que l'on recopiera sur la copie.

	étape 1	étape 2	étape 3	étape 4	étape 5
a	0				
b	1				
$b-a$					
m					

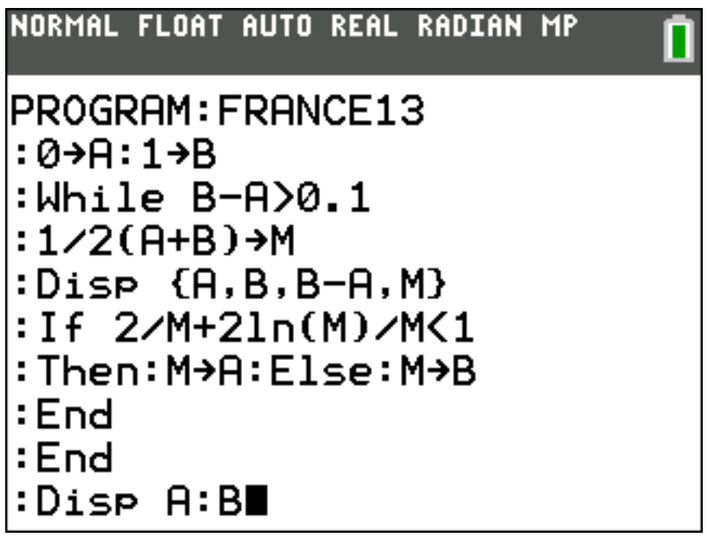
b. Que représentent les valeurs affichées par cet algorithme ?

c. Modifier l'algorithme ci-dessus pour qu'il affiche les deux bornes d'un encadrement de β d'amplitude 10^{-1}.

Question 4)a):
On nous demande donc une sorte de trace de l'algorithme, avec les états des variables lors des différentes itérations de la boucle.

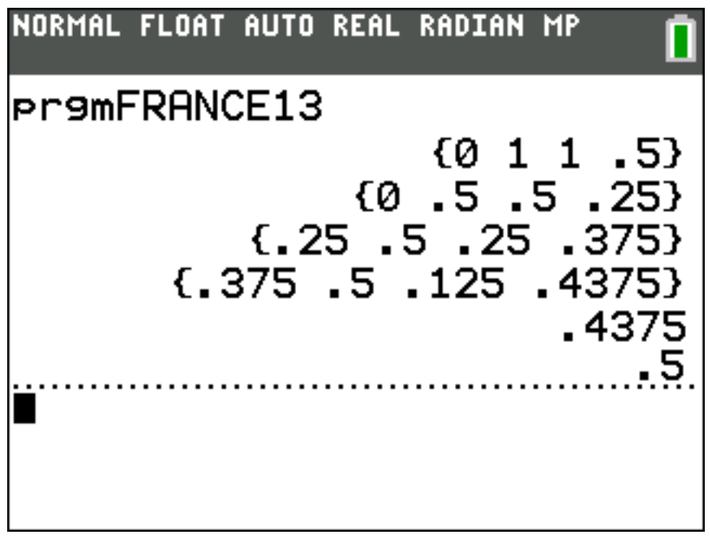
Selon une astuce rappelée hier soir, il suffit de prendre notre calculatrice TI-76/82/83/84, d'y programmer l'algorithme en question, et de rajouter une instruction de sortie dans la boucle.

L'énoncé donnant le début de la trace, on peut déterminer l'endroit exact où placer cet affichage.
Il doit donc se faire avant les modifications de A et B (A et B valant toujours leurs valeurs d'initialisation 0 et 1 à la 1ère étape), mais après l'affectation de M qui n'aurait pas de valeur sinon.



On obtient alors directement sur l'écran de la calculatrice les résultats à recopier:

Pack et mise en page par Laurae, rédigé par critor sur le forum, administrateurs de TI-Planet.org



L'étape 5 ne correspond pas à un passage dans la boucle, donc son détail semble discutable. M n'est en effet pas modifié.

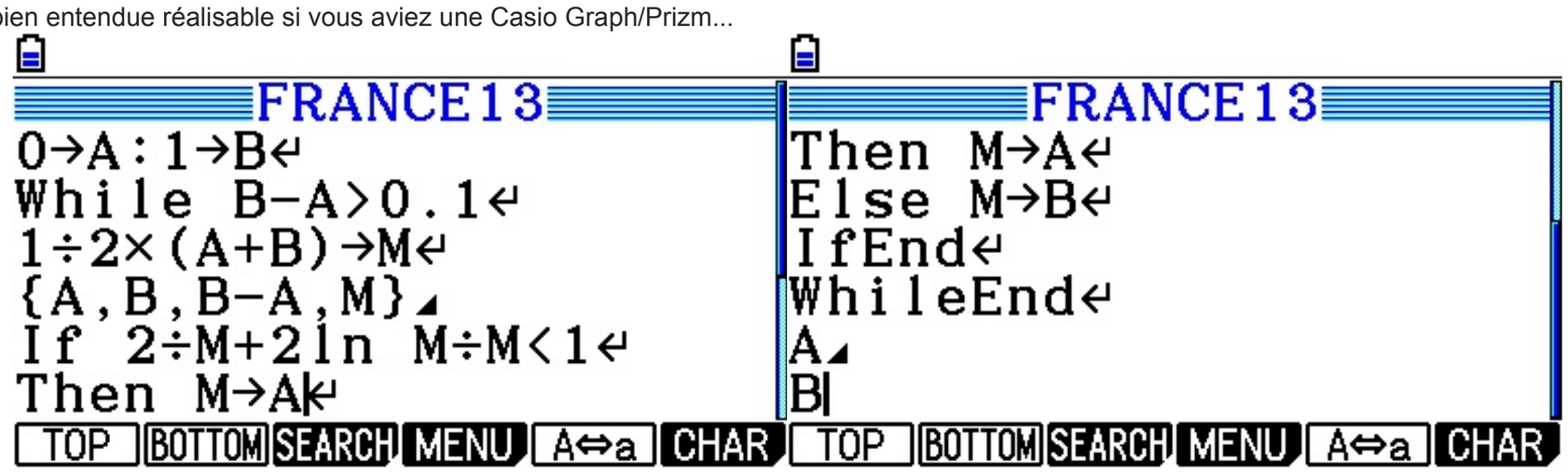
Mais ceux qui n'ont pas fait sur machine et ont déroulé l'algorithme de dichotomie à la main auront peut-être mis une valeur différente.

Pour ma part, je réponds exactement à la question de l'énoncé en ne modifiant pas la valeur de M entre les étapes 4 et 5, c'est-à-dire en donnant sa vraie valeur.

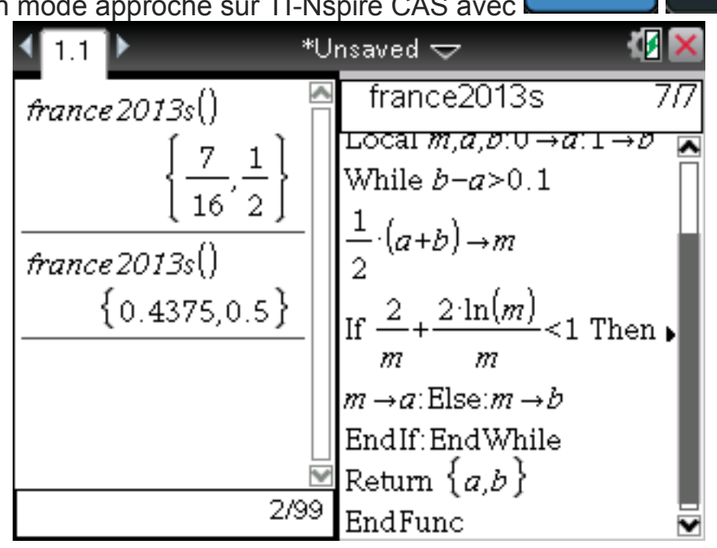
	étape 1	étape 2	étape 3	étape 4	étape 5
a	0	0	0,25	0,375	0,4375
b	1	0,5	0,5	0,5	0,5
b-a	1	0,5	0,25	0,125	0,0625
m	0,5	0,25	0,375	0,4375	0,4375

Il est à mon avis fort possible que les deux possibilités soient acceptés dans les consignes de correction pour la 5ème étape.

La même chose reste bien entendue réalisable si vous aviez une Casio Graph/Prizm...



... ou encore sur TI-Nspire, et il ne faut pas oublier d'évaluer en mode approché sur TI-Nspire CAS avec ctrl ≈ enter



Question 4)b):
Cet algorithme recherche par dichotomie sur l'intervalle [a,b]=[0,1], un encadrement d'amplitude au plus $0,1=10^{-1}$ (condition de poursuite de la boucle tant que) de α tel que f(α)=1.
C'est cet encadrement [a,b] qui est affiché à la fin et vaut ici [0,4375;0,5].

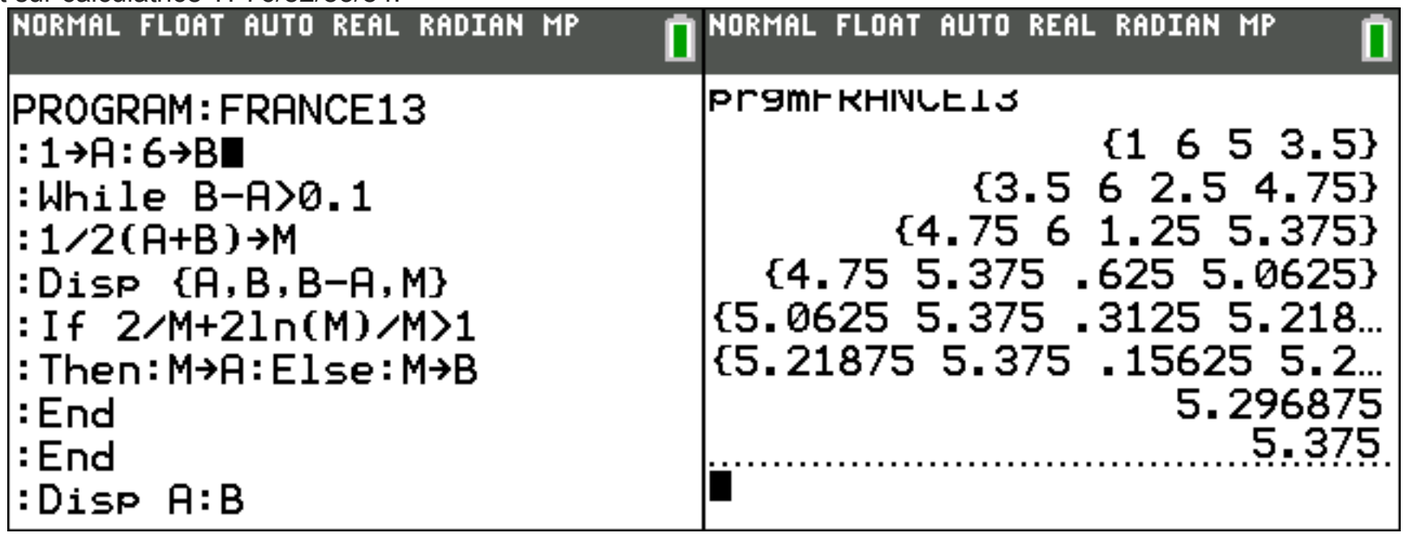
Question 4)c):
La 2ème solution β de l'équation f(x)=1, ne se trouve pas sur l'intervalle [0,1], mais sur [1,+∞[.
Pour la rechercher, il suffit donc de modifier l'intervalle [a,b] de départ, de [0,1] à par exemple [1,6], et de tenir compte du fait que f est décroissante sur cet intervalle.

Pack et mise en page par Laurae, rédigé par critor sur le forum, administrateurs de TI-Planet.org

Voici l'algorithme avec les modifications indiquées:

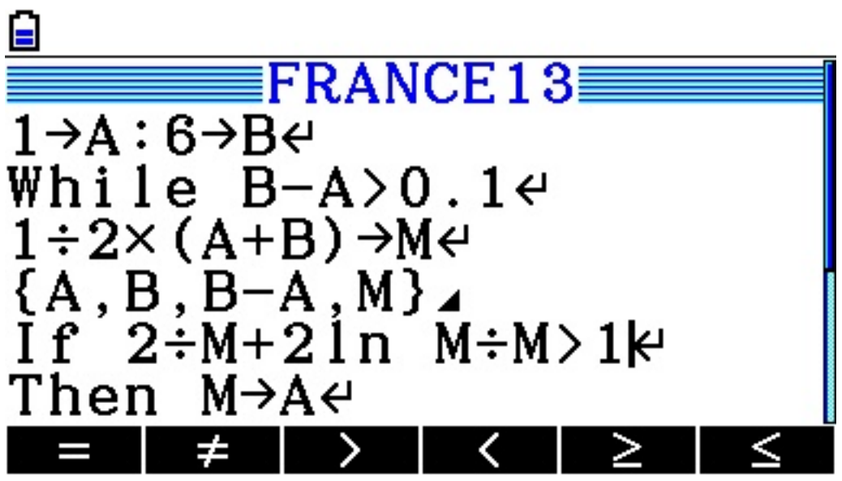
```
Variables:
   a,b et m sont des réels
Initiatlisation:
   Affecter à a la valeur 1    (*)
   Affecter à b la valeur 6    (*)
Traitement:
   Tant que b-a>0,1
      Affecter à m la valeur 1/2(a+b)
      Si f(m)>1 alors affecter à a la valeur m    (*)
      sinon affecter à b la valeur m
      Fin de Si
   Fin de Tant que
Sortie:
   Afficher a
   Afficher b
```

On confirme le bon fonctionnement sur calculatrice TI-76/82/83/84:



Un encadrement à 10^{-1} près de β est donc [5,296875;5,375].

Voici la même modification pour Casio Graph/Prizm...



... et maintenant pour TI-Nspire: