KL55

CONCOURS ENSAM - ESTP - ECRIN - ARCHIMEDE

Epreuve d'Informatique MP

durée 3 heures

L'usage de la calculatrice est interdit

Indiquez en tête de copie ou de chaque exercice le langage utilisé. On pourra utiliser la notation t[i] pour accéder à l'élément n° i d'une liste t.

1. Chercher-remplacer

Écrire la fonction

```
    annuleNegatifs       donnée-résultat t : liste d'entiers
```

qui recherche dans une liste d'entiers les entiers négatifs et les remplace par des 0.

2. Racine carrée

Écrire la fonction

```
    racineCarree         donnée a : réel
                         résultat : réel
```

qui calcule la racine carrée d'un nombre réel positif a par l'algorithme de Newton. Préciser bien le test d'arrêt de l'algorithme.

Principe : la suite $u_0 = 1,\ u_{n+1} = \dfrac{u_n + \dfrac{a}{u_n}}{2}$ converge vers $\sqrt{a}$

3. Nombre de 1

Écrire la fonction

```
    nombreDeUn          données n : entier positif ou nul
                        résultat : entier
```

qui calcule le nombre de 1 dans l'écriture binaire de *n*.
Par exemple,

```
    nombreDeUn(23)→ 4
```

car 23 s'écrit 10111 en base 2

4. Qu'affichera ce programme ?

```
x ← 1
y ← 1
tant que x ≤ y faire
  afficher (x)
  x ← x * 2
  y ← y + 10
fin tant que
```

5. Le triangle de Pascal

Le triangle de Pascal est un tableau qui se construit de la manière suivante :

```
1
1   1
1   2   1
1   3   3   1
1   4   6   4   1
1   5  10  10   5  1
```

En convenant que les lignes et colonnes sont numérotées à partir de 1, la ligne *i* contient *i* éléments (numérotés de 1 à *i*). L'élément 1 et l'élément *i* valent 1. L'élément *j* (avec $1<j<i$) est égal à la somme des éléments n° *j* et n° *j*-1 de la ligne précédente.

Écrire la fonction suivante :

```
    lignePascal         données n :entier
                        résultat t : liste d'entiers
```

qui range dans la liste *t* la ligne *n* du triangle de Pascal, et ce sans utiliser de matrice ou de tableau à deux dimensions.

6. Tri d'une liste à petit ensemble de valeurs

Le but de cet exercice est de trier une liste t dont les éléments ne peuvent prendre qu'un "petit" nombre fini de valeurs (ici, trois valeurs : 0, 1 ou 2).

Par exemple, de la liste initiale :

2	0	1	0	2	1	1	0	1	2	2	1	0
1	2	3	4	5	6	7	8	9	10	11	12	13

il faut obtenir la liste finale :

0	0	0	0	1	1	1	1	1	2	2	2	2
1	2	3	4	5	6	7	8	9	10	11	12	13

Le programme sera itératif. Pour trouver comment écrire le corps de la boucle, on suppose que la liste t (de taille n) a été traitée jusqu'au rang i, et que sont connus j et k vérifiant
- tous les éléments du tableau d'indice inférieur ou égal à j sont égaux à 0
- tous les éléments d'indice supérieur à j et inférieur ou égal à k sont égaux à 1
- tous les éléments d'indice supérieur à k et inférieur ou égal à i sont égaux à 2.

0 ... 0	1 ... 1	2 ... 2	x	......
1 j	j+1 k	k+1 i	i+1	n

Le prochain élément à traiter est $t[i+1]$ noté x.

- En supposant $1 \leq j < k < i < n$ (c'est à dire qu'il y a au moins un 0, un 1 et un 2 placés), quelles sont les modifications à faire sur t, i, j et k si $x=2$? si $x=1$? si x=0?

Il se peut qu'il n'y ait pas encore de 0, 1 ou 2 dans la partie de liste déjà triée. Il faut en tenir compte dans les modifications de t, i, j et k selon les valeurs de x.

- Écrire le traitement complet de tri.

7. Génération de nombres pseudo-aléatoires

Dans son livre *Seminumerical algorithms, The Art of Computer Programming*, vol 2, (1969), Donald E. Knuth présente un algorithme de génération de nombres pseudo-aléatoires, basé sur la suite

$$a_{n+1}=f(a_n)$$

où a_0 est un nombre entier positif inférieur à 10 000 000 000 et f une fonction entière à valeur entière de [0..9 999 999 999] dans [0..9 999 999 999]. Le calcul de $f(x)$ emploie 12 étapes différentes, répétées un nombre variable de fois, selon la valeur de x.
Ces 12 étapes sont les suivantes :

K1 : prendre le chiffre des milliards de x

```
K1(3 456 789 123)→3,    K1(2004)→0
```

K2 : prendre le chiffre des centaines de millions de x.

```
K2(3 456 789 123)→4,    K2(2004)→0
```

K3 : si x<5 000 000 000, ajouter 5 000 000 000 à x

```
K3(3 456 789 123)→3 456 789 123,   K3(2004)→5 000 002 004
```

K4 : élever x au carré, supprimer les cinq derniers chiffres, prendre les 10 derniers chiffres du résultat.

```
K4(3 456 789 123)→3 910 408 911,  K4(2004)→40
```

K5 : multiplier x par 1 001 001 001 et prendre les 10 derniers chiffres du résultat

```
K5(3 456 789 123)→2 368 912 123,  K5(2004)→6 006 006 004
```

K6 : si x<100 000 000, ajouter 9 814 055 677 à x, sinon ôter x de 10 000 000 000

```
K6(3 456 789 123)→6 543 210 877,  K6(2004)→9 814 057 681
```

K7 : permuter les 5 premiers chiffres de *x* avec les 5 derniers

```
K7(3 456 789 123)→8 912 334 567,  K7(2004)→200 400 000
```

K8 : multiplier *x* par 1 001 001 001 et prendre les 10 derniers chiffres du résultat (= K5)

```
K8(3 456 789 123)→2 368 912 123,  K8(2004)→6 006 006 004
```

K9 : alors enlever 1 à tous les chiffres non nuls de *x*

```
K9(3 456 789 123)→2 345 678 012,  K8(2004)→1 003
```

K10 : si *x*<100 000 alors élever *x* au carré et lui ajouter 99 999 sinon ôter 99 999 de *x*

```
K10(3 456 789 123)→3 456 689 124, K10(2004)→4 116 015
```

K11 : compléter *x* par des 0 à droite pour qu'il ait 10 chiffres

```
K11(3 456 789 123)→3 456 789 123, K11(2004)→2 004 000 000
```

K12 : multiplier *x* par *x*-1, supprimer les cinq derniers chiffres, prendre les 10 derniers chiffres du résultat

```
K12(3 456 789 123)→3 910 374 343, K12(2004)→40
```

L'algorithme K est le suivant :

```
Donnée x : entier
Résultat r : entier
début K
 a←K1(x)
 répéter a+1 fois
  b←K2(x)
  si b=0 alors r←K12(K11(K10(K9(K8(K7(K6(K5(K4(K3(x)))))))))) fin si
  si b=1 alors r←K12(K11(K10(K9(K8(K7(K6(K5(K4(x))))))))) fin si
  si b=2 alors r←K12(K11(K10(K9(K8(K7(K6(K5(x)))))))) fin si
  si b=3 alors r←K12(K11(K10(K9(K8(K7(K6(x))))))) fin si
  si b=4 alors r←K12(K11(K10(K9(K8(K7(x)))))) fin si
  si b=5 alors r←K12(K11(K10(K9(K8(x))))) fin si
  si b=6 alors r←K12(K11(K10(K9(x)))) fin si
  si b=7 alors r←K12(K11(K10(x))) fin si
  si b=8 alors r←K12(K11(x)) fin si
  si b=9 alors r←K12(x) fin si
 fin répéter
 retour r
fin K
```

- Écrire les fonctions K1, K2, K3, K4, K5, K6, K7, K8, K9, K10, K11, K12

- Écrire la fonction K

Note : on supposera que les entiers sont représentables sans limite de valeur.

8. Programmation d'expressions logiques

Les calculs des valeurs de vérité d'expressions comme $\forall x \in E\ P(x)$ ou $\exists x \in E\ P(x)$ peuvent être programmés si *E* est un sous-ensemble **fini** de **N** et *P* une expression logique calculable pour toute valeur de *E*.

Programmer le calcul de la valeur de vérité des expressions suivantes :

- $\forall x \in [0..1000]$ $(x^2 \text{ modulo } 4) \leq 1$
- $\forall n \in [1..1000]$ $\exists x \in [n..2n]$ $\forall y \in [2..x-1]$ $(x \text{ modulo } y) \neq 0$