Algorithmique et programmation

Exercice 1 (Invariant de boucle)**.** On voudrait savoir ce que fait l'algorithme suivant sur le tableau `T` dont les indices de début et de fin sont respectivement `premier` et `dernier`.

```
j := premier
pour i de premier à dernier - 1
        si T[i] <= T[dernier] alors
            echanger T[i] et T[j]
            j := j + 1
        fin si
fin pour
echanger T[dernier] et T[j]
```

Pour cela on va utiliser les invariants de boucle.
Question 1 Pour avoir une idée du déroulement de l'algorithme faites le tourner sur le tableau suivant avec `premier` valant 0 et `dernier` valant 7.

2	9	3	8	6	10	4	5

Question 2 Soit la propriété $P(i, j)$: "*tous les termes d'indice compris entre* 0 *et* $j - 1$ *sont plus petits que* `T[dernier]` *alors que ceux d'indice compris entre* j *et* $i - 1$ *sont plus grands.*" Montrez que $P(0, 0)$ est vraie. Montrez par récurrence que $P(i, j)$ est vraie à l'entrée de la boucle i.
Question 3 Indiquez ce que fait le programme.

Exercice 2 (Produit de matrices)**.** Soit A et B deux matrices carrées de dimension n. (N.B. On appelle matrice carrée de dimension n un tableau de flottants à deux dimensions de taille $n \times n$). On note $A[i][j]$ le coefficient (d'indice i,j) de la matrice A situé sur la ligne i et la colonne j.

Le produit $C = A \times B$ de deux matrices carrées A et B de dimension n est une matrice carrée de dimension n dont le terme d'indice i,j est calculé comme la somme des produits terme à terme des coefficients de la ligne i par ceux de la colonne j. C'est à dire (sous forme mathématique) par

$$C[i][j] = \sum_{k=0}^{n-1} A[i][k] * B[k][j].$$

Par exemple, si A et B sont des matrices carrées de taille 3 alors

$$C[i][j] = A[i][0] * B[0][j] + A[i][1] * B[1][j] + A[i][2] * B[2][j]$$

Question 1 Ecrivez la fonction `terme_produit` qui renvoie le coefficient d'indice i,j du produit de la matrice A par la matrice B. Le prototype est

```
float terme_produit(int i, int j, int n, float **A, float **B);
```

Question 2 Ecrivez le code de la fonction qui crée la matrice produit, calcule tous les termes de la matrice produit de A et de B et renvoie l'adresse de l'espace créé. Le prototype est

```
float ** produit_mat(int n,float **A, float **B);
```

Question 3 Refaites les questions 1 et 2 en codant les matrices comme des tableaux à 2 dimensions, dans le style du C ansi, et non plus comme des tableaux de pointeurs sur des tableaux, dans le style du C K&R. Les prototypes sont

```
float terme_produit2(int i, int j, int n, float A[][n], float B[][n]);
void* produit_mat2(int n, float A[][n], float B[][n]);
```

Question 4 Intégrez les 4 fonctions précédentes dans le programme suivant :

```
int main()
{ float t[]={1,2,3,4,5,6,7,8,9}, *A[3]={t,t+3,t+6}, **C=produit_mat(3,A,A);
  float D[][3]={{1,2,3}, {4,5,6}, {7,8,9}}, (*E)[3]=produit_mat2(3,D,D);
  affm1(3,A); affm1(3,C); freem1(C);
  affm2(3,D); affm2(3,E); freem2(E);
  getchar();
  return 0;
}
```

Exercice 3 (Fraction continue)**.** La décomposition en fraction continue du nombre α est une expression $\alpha = a_0 + \cfrac{1}{a_1+\cfrac{1}{a_2+\cfrac{1}{a_3+\cfrac{1}{a_4+\cfrac{1}{\ddots}}}}}$ où tous les termes a_0, a_1, ..., a_i, ... sont entiers et les dénominateurs de toutes les fractions sont positifs. Donc ces termes peuvent se calculer comme $a_i = \lfloor u_i \rfloor$, la suite u étant définie par $u_0 = \alpha$ et $1/u_{i+1} = u_i - a_i$.

Si α est rationnel alors la suite est finie. Le dernier terme étant $a_n = u_n$. On peut poser $a_i = u_i = \infty$ pour $i > n$.

Question 1 Ecrivez la fonction `decomp` dont le prototype est

```
void decomp(double alpha, int n);
```

qui affiche les n premiers coefficients (les termes a_n) de la décomposition en fraction continue du nombre α. Si la suite est trop courte, les ∞ qui suivent ne seront pas affichés.

Question 2 Ecrivez (en modifiant légèrement votre fonction précédente) la fonction `decomp_tab` qui remplit un tableau avec les n premiers coefficients de la décomposition en fraction continue du nombre α, en complétant par des 0 si la suite est trop courte. Ce tableau sera supposé créé à la bonne taille. Le prototype est

```
void decomp_tab(double alpha, int n, int *tableau);
```

On appelle convergent d'ordre k d'un nombre α, noté $r_k(\alpha)$ le nombre rationnel obtenu en utilisant les k premiers coefficients de la décomposition en fraction continue de α.

$$r_k(\alpha) = a_0 + \cfrac{1}{a_1+\cfrac{1}{a_2+\cfrac{1}{\ddots\,{}_{a_{k-2}+\frac{1}{a_{k-1}}}}}}$$

On peut le calculer en initialisant $r := \infty$ puis pour $i = k-1,\ k-2,\ \ldots,\ 1,\ 0$ on fait $r := a_i + 1/r$.

Question 3 Ecrivez la fonction `convergent` qui renvoie la valeur approchée du convergent d'ordre k à partir d'une décomposition en fraction continue d'ordre supérieur à k du nombre α rangée dans un tableau. Le prototype est

```
double convergent(int k, int *tableau);
```

On remplacera évidemment ∞ par 0. Donc on initialisera $r := 0$ puis on remplacera $1/0$ par 0. La boucle contiendra `r=tableau[i]+(r?1/r:0);`

Question 4 Soit $\pi = 3,141592653589$, calculez la décomposition en fraction continue d'ordre 6 et affichez les 5 premiers convergents.

Question 5 Dans la question 4 on calcule les n premiers convergents avec un temps en n^2. Refaites la question 4 avec un temps en $O(n)$ en remarquant que $r_i(\alpha) = v_i/w_i$ avec $v_{-2} = 0$, $v_{-1} = 1$, $v_i = a_i v_{i-1} + v_{i-2}$ et $w_{-2} = 1$, $w_{-1} = 0$, $w_i = a_i w_{i-1} + w_{i-2}$.

Question 6 On peut éviter les pertes de précision des questions 1 et 2 lors du calcul de l'inverse d'un nombre réel, en appliquant plutôt l'algorithme d'Euclide à α et 1 : $a_i = \lfloor u'_{i-1}/u'_i \rfloor$ avec $u'_{-1} = \alpha$, $u'_0 = 1$, $u'_{i+1} = u'_{i-1} - a_i u'_i$. Complétez la fonction :

```
void decomp2(long double alpha);
```

Exécutez `decomp2(M_PI); decomp2(2*acosl(0));`

Exercice 4 (Cryptage simple)**.** L'algorithme de cryptage de César est un algorithme qui prend une clef k et qui remplace la $n$ième lettre de l'alphabet dans le message en clair par la $n+k$ième modulo 27 (le 27ème caractère est l'espace) dans le message crypté. Modulo signifiant que si on dépasse la limite supérieure des lettres, on reprend au début de l'alphabet.

Par exemple si la clef est $k = 4$ alors la lettre a est remplacée par la lettre e et la lettre z par c.

Ecrivez la fonction `decrypt(char *c,j)` qui prend en entrée un tableau de caractères de taille j qui est la phrase cryptée et qui affiche les 26 possibilités de phrases en clair.