

Examen de Langage C (V. Granet)

Durée : 2h

Aucun document autorisé

Mobiles interdits

Notez que les affirmations (antécédents, conséquents, rôles, et invariants) dans vos codes C entreront pour partie dans la note finale.

- 1. Deux entiers naturels sont des nombres amiables si chacun est égal à la somme des diviseurs de l'autre. Par exemple, 220 et 284 sont des nombres amiables. Écrivez une fonction qui teste si deux nombres sont amiables ou pas. La fonction renvoie un booléen (0 ou $\neq 0$).

Question sur 5 pts :

```
/*
 * Antécédent : x>=0
 * Rôle : renvoie la somme des diviseurs de x
 */
int sommeDiv(int x) {
    int som=1, divMax=x/2;
    /* sommer tous les diviseurs de x sur [2;divMax[ */
    for (int i=2; i<=divMax; i++)
        // invariant : som = somme des k premiers diviseurs de x sur [2;i[
        if ((x%i)==0) som+=i;
    return som;
}
/*
 * Antécédent : a>=0 et b>=0
 * Rôle : renvoie 1 si a et b sont deux nombres amiables et 0 sinon
 */
int etesVousAmiables(int a, int b) {
    assert(a>=0 && b>=0);
    return sommeDiv(a)==b && a==sommeDiv(b);
}
```

- 2. Écrivez la fonction `quelleLigne` qui renvoie l'indice de la première ligne d'une matrice $M \times N$ dont la somme des éléments est comprise dans un intervalle $[a; b]$. La matrice et les bornes de l'intervalle sont passées en paramètre. Si une telle ligne n'existe pas, la fonction renvoie -1 .

Question sur 5 pts :

```
/*
 * Antécédent : a<=b
 * Rôle : renvoie l'indice la première ligne de la matrice
 *         mat dont la somme est dans l'intervalle [a;b]
 */
int quelleLigne(double mat[M][N], double a, double b) {
    assert(a<=b);
    for (int i=0; i<M; i++) {
        /* calculer la somme des valeurs de la ligne i */
        double somme=0.0;
        for (int j=0; j<N; j++)
            somme+=mat[i][j];
        /* somme dans [a;b] ? */
        if (somme>=a && somme<=b)
            return i;
    }
    /* aucune ligne dont la somme les valeurs dans [a;b] */
    return -1;
}
```

- 3. Un polynôme $p(x)$ est représenté par un tableau tel que l'élément d'indice i est égal au coefficient de x^i . En utilisant les déclarations suivantes :

```
#define DEGREMAX 40
typedef double polynome[DEGREMAX];
```

Écrivez la fonction `eval` qui prend en paramètre un polynôme p et un réel x qui renvoie l'évaluation de $p(x)$ selon la méthode de Hörner :

$$p(x) = (((\dots(p[n] \times x + p[n-1]) \times x + \dots + p[1]) \times x + p[0])$$

Question sur 5 pts :

```
/*
 * Rôle : renvoie l'évaluation de p(x)
 * Note : Algorithme d'Horner
 */
double evaluer(polynome p, double x) {
    int n = DEGREMAX-1;
    double valeur = p[n];
    /* valeur = \sum_{k=n} p[k] x^{k-1} = p[n] */
    for (int d=n-1; d >= 0; d--) {
        /* valeur = \sum_{k=n} p[k] x^{k-1} */
        valeur = valeur*x + p[d];
    }
    /* valeur = \sum_{k=n} p[k] x^k */
    return valeur;
}
```

- 4. Écrivez le programme `head` qui affiche les 10 premières lignes de l'entrée standard. Attention, le nombre de lignes existant sur le fichier d'entrée est quelconque, et peut contenir moins de 10 lignes. Par exemple, pour afficher les 10 premières du fichier `fich.txt`, on exécutera le programme `head` comme suit : `head < fich.txt`.

Question sur 5 pts :

```
#include <stdio.h>
#include <stdlib.h>

int main(void) {
    int c, nbLignes = 10; // par défaut
    /* afficher les nbLignes premières lignes */
    /* on s'arrête si EOF atteinte avant */
    while (nbLignes>0)
        switch (c=getchar()) {
            case EOF : return EXIT_SUCCESS;
            case '\n' : nbLignes--;
            default : putchar(c);
        }
    return EXIT_SUCCESS;
}
```