CRO - 3TC - examen 2018-20189

Durée : 2h heures, tous documents autorisés

Remarques :

- Le barème est donné à titre indicatif (il pourra être modifié éventuellement).
- Pour évaluer les programmes C, on prendra en compte (dans l'ordre décroissant d'importance) : la validité de l'algorithme, la validité syntaxique du programme, la clarté du programme et les commentaires, la forme générale de la présentation.

1 Compilation (3 pts)

1. Expliquer (en moins de 10 lignes) la différence entre un fichier objet et un fichier exécutable.
2. Écrire le fichier Makefile pour le projet C comportant les fichiers suivants :
   - deux fichiers de fonctions C : `note-CRO.c` et `note-ARC.c` (ainsi que leur fichiers `.h` associés)
   - Un fichier principal : `NOTE-3TC.c` contenant une fonction `main()`

2 Quelques exercices (8pts)

2.1 PGCD

Ecrire un fonction :
`int pgcd(int a,int b)`
qui calcule le plus grand commun diviseur de deux entiers positifs. On pourra utiliser l'algorithme de son choix. Si vous n'en connaissez pas, l'algorithme d'euclide vous est rappelé ci-contre. (`mod`) est l'opérateur modulo (`%` en C).

```
int pgcd(int a, int b)
    SI (a>b) ALORS
        SI (a mod b == 0) ALORS
            res ← b
        SINON
            res ← PGCD(b, a mod b)
        FINSI
    SINON
        SI (b mod a == 0) ALORS
            res ← a
        SINON
            res ← PGCD(a, b mod a)
        FINSI
    FINSI
    RETURN res;
```

2.2 Palindrome

Ecrire une fonction :
`int Palindrome (char T[], int N)`

qui dit si la chaîne de caractère contenue dans un tableau de caractère `T` de taille `N` (i.e tous les caractères jusqu'au caractère \0), forment un palindrome. Un palindrome est une phrase qui se lit de la même façon dans les deux sens (exemple : « elle », « radar », « tu l'as trop ecrase Cesar ce port salut » « la mariee ira mal »). Notez que les espaces ne compte pas....

2.3 Coefficient binomial

Le coefficient binomial des entiers naturel n et k noté $\left(\begin{array}{c} n \\ k \end{array} \right)$ (anciennement C_n^k : combinaison de k parmi n) peut être calculé récursivement avec les informations suivantes :

$$\left(\begin{array}{c} n \\ k \end{array} \right) \text{ vaut 1 si } k = n \text{ ou } k = 0$$
$$\left(\begin{array}{c} n \\ k \end{array} \right) + \left(\begin{array}{c} n \\ k+1 \end{array} \right) = \left(\begin{array}{c} n+1 \\ k+1 \end{array} \right) \text{ si 0<k<n}$$

Écrire une fonction :

`int cnp (int n, int k)`

qui, par une double récursion, calcule $\left(\begin{array}{c} n \\ k \end{array} \right)$,

3 Calcul matriciel 5pts

On veut écrire un programme C qui calcule le produit de deux matrices d'entiers : une matrice A de taille $N \times P$ (N lignes, P colonnes) par une matrice B de taille $P \times M$. On rappelle que le coefficient $c_{i,j}$ de la matrice $C = A \times B$ vaut

$$c_{i,j} = \sum_{k=1}^{P} a_{i,k} b_{k,j}$$

P étant le nombre de colonnes de A et aussi le nombre de lignes de B, La matrice C étant de taille $N \times M$

3.1 Produit de matrice, version statique

On supposera d'abord que les trois valeurs N, M et P sont connues à la compilation et sont définies au début du fichier grâce à des `#define`. Par exemple :

```
#define N 10
#define M 100
#define P 50
```

On supposera aussi que les matrice A et B sont déjà déclarées dans des tableaux à deux dimensions déclarés de la manière suivante, et remplies avec des valeurs.

```
int A[N][P];
int B[P][M];
```

Enfin pour cette première fonction, on supposera que la matrice C a aussi été déclarée dans le programme principal de la fonction suivante, et contient des valeurs aléatoire (i.e. non forcément initialisée à 0) :

```
int C[N][M];
```

Écrire une fonction C :
`void produit_matrice1(int A[N][P], int B[P][M], int C[N][M])`
Qui calcule dans `C` le produit de matrice `A`×`B`. On rappelle que comme `C` est un tableau, le fait de modifier les élément de `C` dans la fonction modifiera effectivement `C` dans la fonction principale. C'est le seul cas en C ou l'on peut modifier l'argument de la fonction. Ici, les macros `N`,`M` et `P` sont définies globalement donc peuvent être utilisées directement dans la fonction sans être passées en paramêtre.

3.2 Produit de matrice, version dynamique

Maintenant on ne suppose pas que les valeurs N, M et P sont connues à la compilation. Pour stocker des matrices dont les tailles ne sont pas connues à la compilation, nous devons allouer dynamiquement des tableaux de tableaux et les considérer comme des pointeurs de pointeurs.

Par exemple on déclarera `A` de la façon suivante : `int **A`. `A` sera d'abord considéré comme un tableau de taille `N` d'éléments de type "`int *`", puis chaque case `A[i]` sera un pointeur vers une zone mémoire de taille `P` d'éléments de type "`int`". De cette manière on accède bien à l'élément $a_{i,j}$ par "`A[i][j]`".

La nouvelle fonction de produit de matrice passera les valeurs des taille `N, M` et `P` en paramêtre et allouera dynamiquement la matrice résultante.

Écrire une fonction C :
`int **produit_matrice2(int **A, int **B, int N, int M, int P)`
qui *renvoit comme résultat* la matrice `C=A`×`B` qu'elle aura préalablement alouée dynamiquement.

4 Arbre binaire de recherche (4pts)

On veut manipuler des arbres binaires triés (ou arbres binaires de recherche, ABR). Les ABR sont des arbres binaires de valeurs avec les propriétés :

— Tous les nœuds du sous-arbre gauche d'un nœud ont une valeur inférieure (ou égale) à la sienne
— Tous les nœuds du sous-arbre droit d'un nœud ont une valeur supérieure (ou égale) à la sienne

On utilise la structure de donnée suivante pour représenter un ABR, chaque nœud stockant un entier :

```
struct model_noeud
{
  int val;
  struct model_noeud *filsGauche;
  struct model_noeud *filsDroit;
} ;

typedef struct model_noeud NOEUD;

typedef NOEUD *ARBRE;
```

1. Écrivez une fonction `rechercheABR` :
   `int rechercheABR(ARBRE root,int valeur)`
   qui recherche la valeur dans l'ABR `root`, renvoie 1 si elle est présente dans l'ABR, 0 sinon.

bonus Écrivez une fonction `insertionABR` :
`ARBRE insertionABR(ARBRE root,int valeur)`
qui insère un nouvel entier `valeur` dans un arbre binaire trié. L'arbre réultat de la fonction est toujours un ABR.