UNIVERSITÉ PARIS SUD

Examen final
Cours d'analyse, algorithmique
—Master 2 CCI—

Les documents manuscrits, sujets de travaux pratiques et dirigés ainsi que les supports de cours sont autorisés. Tous les autres documents tels que livres, calculatrices, téléphones portables et ordinateurs sont interdits.

Les exercices sont indépendants. Si l'on ne sait pas justifier une question, on peut néanmoins utiliser la réponse dans la suite.

Durée : 3 heures

▸ Exercice 1. (Ensembles codés par tableaux triés)

Dans cet exercice, on code les ensembles de nombres à l'aide de tableaux sans répétitions. Dans un premier temps, on ne suppose pas que les tableaux sont triés. Par exemple l'ensemble $A := \{1, 4, 6, 7\}$ pourra être encodé par le tableau `tabA` $:= [4, 6, 1, 7]$ de taille `tailleA` $:= 4$.

1. Étant donnés un tableau `tabA` de taille `tailleA` encodant l'ensemble A et un élément e, écrire un algorithme qui répond si e appartient à l'ensemble A.
2. En déduire un algorithme qui prend en paramètre deux tableaux :
   – `tabA` de taille `tailleA` encodant un ensemble A
   – `tabB` de taille `tailleB` encodant un ensemble B
   et qui remplit un tableau `tabC` de taille `tailleC` pour coder la **la différence ensembliste** $C := A - B$, c'est-à-dire l'ensemble des éléments de A qui ne sont pas dans B. Note : on pourra supposer que `tabC` est alloué avec une taille suffisante, et on ne demande pas d'écrire cette allocation.
3. Montrer que la complexité de l'algorithme obtenu est dans le pire des cas

$$O(\texttt{tailleA} \times \texttt{tailleB})\,.$$

Dans cette deuxième partie, on suppose que les tableaux `tabA` et `tabB` sont triés.

4. En s'inpirant de l'algorithme de fusion de deux tableaux, écrire un algorithme qui remplit le tableau `tabC` de taille `tailleC` pour coder la différence $C := A - B$.
5. Montrer que la complexité de l'algorithme est dans le pire des cas

$$O(\texttt{tailleA} + \texttt{tailleB})\,.$$

▸ **Exercice 2. (Recherches)**

Soit T un tableau de n nombres réels (`float`). Étant donné un intervalle $[a, b]$ (avec $a < b$), on veut savoir s'il existe, dans le tableau T, un nombre qui appartient à l'intervalle $[a, b]$.

1. Donner un algorithme qui
   - retourne un nombre x dans l'intervalle, ainsi que sa position dans le tableau si un tel nombre existe. Si plusieurs nombres conviennent, on pourra retourner celui que l'on veut.
   - retourne `NonTrouvé` sinon.

   Par exemple, pour le tableau $(1.1, 7.2, 3.5, 1.2, 10.1)$ avec $[a, b] = [2.0, 4.0]$, on retournera $(3.5, 2)$. Sur le même tableau si $[a, b] = [2.0, 3.0]$, on retournera `NonTrouvé`.
2. Quelle est la complexité de cet algorithme ?

On suppose maintenant le tableau trié.

3. Par recherche dichotomique, donner un algorithme de type « diviser pour régner » qui effectue le même travail.
4. Quelle est la complexité de ce nouvel algorithme ?

▸ **Exercice 3. (Dessin de Carré)**

Écrire un programme qui, pour un entier positif l donné, affiche un carré dont le côté est de longueur L. Sur les lignes horizontales, les «*» sont séparées par des espaces.

Un exemple : lorsque l vaut 5, l'affichage est

```
* * * * *
*       *
*       *
*       *
* * * * *
```

L'affichage doit bien entendu se faire ligne par ligne. On pourra utiliser les commandes `Affiche("*")` et `Affiche("␣")` pour afficher une étoile ou un espace et `NouvelleLigne` pour aller à la ligne.

▸ **Exercice 4. (Fonction de MacCarthy)**

On considère la fonction suivante :

```
int mccarthy(int n)
{
  if (n > 100) return n-10;
  else return mccarthy (mccarthy (n + 11));
}
```

1. Donner la suite des appels récursifs ainsi que la valeur de retour pour l'appel `mccarthy(94)`. Indication : il y a 15 appels.

▸ **Exercice 5. (Retournement d'un tableau)**

Écrire une fonction `retourne` qui prend en paramètre T et n où T est un tableau d'entiers de taille n et qui retourne le tableau en place. Par exemple, si l'on donne en entrée le tableau de taille 5 suivant

0	1	2	3	4
1	5	3	6	2

on doit avoir en sortie le tableau

0	1	2	3	4
2	6	3	5	1