| | | |
|---|---|---|
| université de BORDEAUX | **ANNEE UNIVERSITAIRE 2015 / 2016**<br>SESSION 1 DE PRINTEMPS<br><br>**PARCOURS : L1 Maths/Info** **Code UE : J1MI2013**<br><br>**Date : 18 mai 2016** **Heure : 8h30** **Durée :** 1h30<br>**Documents :** non autorisés.<br>**Epreuve de Mme :** C. Blanc.<br><br>**Consignes :** Vous devez répondre directement sur le sujet qui comporte 8 pages. Insérez ensuite votre réponse dans une copie d'examen comportant tous les renseignements administratifs. | Collège Sciences et Technologies |

Numéro d'anonymat :

**Questions de cours : (3 points)**

1. Écrire la fonction `echanger` qui échange le contenu de deux cases d'un tableau d'entiers. Quelle est sa complexité ?

2. Lorsque pour trier un tableau d'entiers en ordre croissant on applique l'algorithme du **tri à bulles** et qu'on arrête l'exécution de l'algorithme après `k` itérations `(1 <= k < n)`, combien d'éléments se trouveront à leur place définitive (et ne seront plus considérés si on reprend l'exécution) ? Justifiez votre réponse.

3. Quand on utilise l'algorithme de recherche dichotomique combien d'étapes au maximum sont nécessaires pour trouver un entier `x` parmi `1000` entiers ordonnés ? Justifiez.

**Exercice 1 : Les booléens. (2,5 points)**

Dans un restaurant les clients peuvent commander une entrée, un plat ou un dessert.
Pour enregistrer les commandes de `n` clients on utilise trois tableaux de `n` booléens :

- `bool E[]` pour stocker les entrées commandées. Si le client `i` commande une entrée `E[i]`sera égal à `true` sinon `E[i]` sera égal à `false`.
- `bool P[]` pour stocker les plats commandés. Si le client `i` commande un plat `P[i]`sera égal à `true` sinon `P[i]` sera égal à `false`.
- `bool D[]` pour stocker les desserts commandés. Si le client `i` commande un dessert `D[i]`sera égal à `true` sinon `D[i]` sera égal à `false`.

La synthèse des commandes se fera dans un tableau `bool C[]` qui sera initialisé par la fonction `commande`.
On suppose que les tableaux. `E`, `P` et `D` sont initialisés avant l'appel de la fonction `commande`.

1. Ecrivez la fonction `void commande(bool C[], bool E[],bool P[], bool D[], int n)` qui remplit le tableau `C` de telle façon que chaque élément `C[i]` soit égal à `true` si le client `i` a commandé une entrée, un plat et un dessert et `false` sinon.

2. Modifiez la fonction `commande` afin que `C[i]` soit égal à `true` si le client `i` a commandé une entrée et un dessert mais **pas** de plat et `false` sinon

3. Modifiez la fonction `commande` afin que `C[i]` soit égal à `true` si le client `i` a commandé exactement deux éléments **dont** un plat et `false` sinon.

Numéro d'anonymat :

**Exercice 2 : Récursivité (2,5 points)**

On considère les fonctions suivantes :

```
bool mystereRec(int t[], int g, int d){
  if (g >= d)
    return true;
  if (t[g]*t[d] < 0 )
    return false;
  return mystereRec(t,g+1,d-1);
}
```

```
bool mystere(int t[], int n){
  return mystereRec(t,0,n-1);
}
```

1. Si `v1=[3,-2,2,1,2,4]`, quel sera le résultat des appels `mystere(v1,6)` ? Justifiez

2. Si `v2=[1,-5,-4,7,-3,0,8]`, quel sera le résultat de l'appel `mystere(v2,7)` ? Justifiez.

3. Si `v3=[1,5,-4,-2,2,8]`, quel sera le résultat de l'appel `mystere(v3,6)` ? Justifiez.

4. Quelle propriété possède un tableau `t` pour lequel `mystere(t,n)` retourne `true` ?

**Exercice 3 : Tri de tableau (4 points)**

1. On applique le tri insertion au tableau `t=[7,1,6,5,3,4,2,8]`. Comment sera modifié `t` après quatre itérations ?

2. Sur le tableau `t` modifié par la question précédente appliquez le tri fusion. Vous détaillerez l'exécution du tri fusion par exemple en dessinant l'arbre des appels.

3. Est-ce que le fait d'appliquer le tri insertion au tableau `t` avant de lui appliquer le tri fusion améliore l'efficacité du tri fusion ? Justifiez.

Numéro d'anonymat :

**Exercice 4 : (8 points).**

On considère la fonction `mystery` suivante :

```
void mystery(int t[], int n, int x){
  int i=0;
  int j=n-1;
  while(i < j){
    if (t[i] == x)
      i=i+1;
    else {
      echanger(t,i,j);
      j=j-1;
   }
}
```

1. Soit `t1=[1,0,2,2,0,2,3,1,3,2]` un tableau de `10` entiers. Détaillez l'application de la fonction `mystery` au tableau `t1` pour `n=10` et `x=2` . Quel sera le contenu de `t1` à la fin de l'exécution de cet appel ?

2. Que fait la fonction `mystery` dans le cas général ?

3. On souhaite écrire une fonction `void separation(int t[], int n, int x)` qui, appliquée à un tableau `t` de `n` entiers, déplace les éléments de `t` de telle façon que :

   - tous les éléments **inférieurs** à `x` sont placés au début du tableau `t`
   - tous les éléments **supérieurs** à `x` sont placés à la fin du tableau `t`
   - entre ces 2 zones on place tous les éléments égaux à `x`.

   Par exemple si `t=[0,4,2,3,1,0,2,1,0,3,4,3,1,2,0]` et `x=3` après l'appel `separation(t,15,3)` on aura `t=[0,0,2,1,0,2,1,0,2,1,3,3,3,4,4]`.

Chaque élément de `t` ne sera observé **qu'une seule fois** au cours de l'exécution de la fonction `separation`.

3.1. Dans quelles conditions la fonction `separation` produira-t-elle **moins** de 3 zones différentes dans le tableau `t` modifié ?

3.2. **Sans écrire de fonctions** décrivez comment la fonction `separation` pourrait être utilisée pour construire un algorithme récursif de tri d'un tableau t d'entiers

Numéro d'anonymat :

3.3. Complétez le code de la fonction `separation` ci-dessous :

```
1. void separation (int t[], int n, int x){
2.     int i,j,k;
3.     i=0;
4.     j=0;
5.     k=n-1;
6.     while (            ){
7.       if (t[j]<x){
8.
9.
10.
11.       }
12.      else{
13.        if (t[j]>x) {
14.
15.
16.         }
17.        else
18.
19.       }
20.    }
21. }
```

3.4. Quelle est la propriété vérifiée par l'entier `i` tout au long de l'exécution de la fonction `separation` ?

3.5. Même question pour `j` ?

3.6. Même question pour `k` ?

FIN