université de BORDEAUX	**ANNEE UNIVERSITAIRE 2017/ 2018** SESSION 2 DE PRINTEMPS **PARCOURS : L1 Maths/Info** **Code UE : 4TPM205** **Date : 18 juin 2018** **Heure : 11h30** **Durée :** 1h30 **Documents :** non autorisés. **Epreuve de Mme :** C. Blanc. **Consignes :** Ce sujet comporte 8 pages d'exercices et 2 pages d'annexe avec les codes des algorithmes de tri de tableau. Vous pouvez détacher l'annexe mais ne séparez pas les autres pages. Vous devez **répondre directement sur le sujet**, insérez ensuite votre réponse dans une copie d'examen comportant tous les renseignements administratifs. **Indiquez votre numéro d'anonymat sur chaque feuille du sujet.**	Collège Sciences et Technologies

Numéro d'anonymat :

Questions de cours :

1. Écrire la fonction `echanger` qui échange le contenu de deux cases d'un tableau t de n entiers. Quelle est sa complexité ?

2. Stabilité des tris :
   a. A quelle condition peut-on dire qu'un algorithme de tri de tableau est stable ?

   b. Question piège : soit t un tableau de n entiers tous distincts dites quels sont les algorithmes de tri vus en cours qui produiront un tri stable sur t.

3. Rappelez la complexité de l'algorithme du tri rapide (quicksort). En justifiant votre réponse dites pourquoi il peut être préféré à l'algorithme du tri fusion pour trier un tableau de taille n.

Exercice 1 : Les booléens.

Dans un jeu vidéo les personnages sont soit des humains (oreilles rondes) soit des extra-terrestres (oreilles pointues). Au cours du jeu chaque personnage peut être contaminé par un virus dans ce cas le personnage aura les yeux rouges, sinon ils sont noirs. Enfin chaque personnage qui est actif plus de vingt jours dans le jeu a le nez bleu, sinon il est rose.

Pour enregistrer l'état des chacun des `n` personnages présents dans le jeu on utilise trois tableaux de `n` booléens :

- le tableau `OP` pour stocker la forme des oreilles. Si le personnage `i` a les oreilles pointues `OP[i]` sera égal à `True` sinon `OP[i]` sera égal à `False`.
- le tableau `YR` pour stocker la contamination des personnages par le virus. Si le personnage `i` a les yeux rouges `YR[i]` sera égal à `True` sinon `YR[i]` sera égal à `False`.
- le tableau `NB` pour stocker le vieillissement des personnages. Si le personnage `i` a le nez bleu `NB[i]` sera égal à `True` sinon `NB[i]` sera égal à `False`.

On suppose que le jeu a commencé depuis plusieurs semaines et on considère que les n personnages présents ont leurs caractéristiques stockées dans les tableaux de booléens `OP`, `YR` et `NB`.

Par exemple : si le personnage N°3 est un **humain contaminé** et **présent dans le jeu depuis 2 jours**, `OP[3]` sera égal à `False` car ses oreilles ne sont pas pointues, `YR[3]` sera égal à `True` car il a les yeux rouges et `NB[3]` sera égal à `False` car son nez n'est pas bleu.

1. En justifiant votre réponse décrivez le personnage N°i pour lequel le booléen `(OP[i]and YR[i]and NB[i])==True`

2. Que peut-on dire du personnage N°i pour lequel le booléen `(OP[i]and YR[i]and NB[i])==False` ?

3. Une potion permet de soigner les extra-terrestres entrés dans le jeu depuis moins de 20 jours ou les humains entrés depuis plus de 20 jours. Ecrivez l'expression booléenne qui vaudra `True` si on peut soigner un personnage malade i.

Numéro d'anonymat :

Exercice 2 : Récursivité

On considère les fonctions suivantes :

```
def enigmaRec(t, i, f, x):

  if (i == f)
    return -1
  if (t[i]==x)
    return i
  return enigmaRec(t,i+1,f,x)
```

```
def enigma(t,n,x):

  return enigmaRec(t,0,n//2,x)
```

1. Si `v1=[5,2,8,4,1,0,7]`, quel sera le résultat des appels `enigma(v1,7,2)` ? Justifiez

2. Quel sera le résultat de l'appel `enigma(v1,7,1)` ? Justifiez.

3. Soit `t` un tableau de `n>=2` entiers, à quelle condition l'appel `enigma(t,n,x)` retournera `-1` ? Justifiez.

4. Quel est l'intérêt que cette fonction retourne un entier plutôt qu'un booléen ?

Exercice 3 : Tri de tableau

1. En décrivant les différentes étapes appliquez le tri insertion sur le tableau `t=[5,8,2,9,1,6,3,7]`.

2. Sur le tableau `t=[1,6,8,11,3,7,15,9]` appliquez le tri fusion . Vous détaillerez l'exécution du tri fusion en dessinant l'arbre des appels et en **indiquant l'ordre des appels**.

3. Si vous ne deviez trier que la moitié d'un tableau de taille n lequel de ces 2 algorithmes choisiriez-vous ? Justifiez votre réponse.

Numéro d'anonymat :

Exercice 4 :

Soit la fonction `mystery1` qui prend en entrée un tableau d'entiers `t` de taille `n` et un tableau d'entiers `d` de taille `p`.

```
1.  def mystery1(t, n, d, p):
2.    for i in range(p):
3.      d[i]=0
4.    for i in range(n):
5.      d[t[i]]=d[t[i]]+1
```

1. Soit `t1=[1,0,2,2,0,2,3,1,3,2]` un tableau de `10` entiers inférieurs à 4. Détaillez l'application de la fonction `mystery1` au tableau `t1` pour `n=10` et `p=4`.

1.1. Quelles valeurs contiendra le tableau `d` à la fin de l'exécution de cet appel ?

1.2. Que représentera le contenu de chaque case du tableau `d` ?

1.3. Quelle propriété doit vérifier p dans le cas général ?

2. On complète la fonction `mystery1` en ajoutant les lignes 6 et 7 pour créer la fonction `mystery2` :

```
1. def mystery2(t, n, d, p):
2.   for i in range(p):
3.      d[i]=0
4.   for i in range(n):
5.      d[t[i]]= d[t[i]]+1
6.   for i in range (1,p):
7.      d[i]=d[i]+d[i-1]
```

Soit `t1=[1,0,2,2,0,2,3,1,3,2]` un tableau de `10` entiers inférieurs à 4. Appliquez la fonction `mystery2` au tableau `t1` pour `n=10` et `p=4` .

2.1. Quelles valeurs contiendra le tableau `d` à la fin de l'exécution de cet appel ?

2.2. Que représentera le contenu de chaque case du tableau `d` ?

3. On souhaite utiliser la fonction `mystery2` pour écrire un algorithme de tri d'un tableau t d'entiers inférieurs à `p`.

3.1. Décrivez le principe de cet algorithme.

Numéro d'anonymat :

3.2. Complétez les lignes 6 et 8 du code de la fonction `tri` ci-dessous :

```
1. def tri (t,n,p) :
2.    d = creerTableau(p)
3.    r = creerTableau(n) #tableau supplémentaire pour faire le tri
4.    mystery2(t,n,d,p)
5.    for i in range(n-1,-1,-1):
6.      k=                    #k est l'indice dans r pour placer t[i]
7.      r[k]=t[i]
8.                            #modifier d[t[i]]
9.    for i in range (n):
10.     t[i]=r[i]
```

3.3. Quelle est la complexité en temps de cet algorithme ? Justifiez.

3.4. Question Bonus : A quelles conditions peut-on utiliser cet algorithme de tri ?

Annexe : t est un tableau d'entiers de taille n. Les algorithmes produisent un tri croissant.

Tri sélection :

```
1  def triSelection (t,n):
2    for  i in range(n) :
3      iMin=i
4      for j in range(i+1,n) :
5        if(t[j]<t[iMin]):
6          iMin=j
7      if(i != iMin):
8        echanger(t,i,iMin)
```

Tri insertion :

```
1  def triInsertion(t, n):
2    for i in range(1,n):
3      e=t[i]
4      j=i-1
5      while (j>=0 and t[j]>e):
6        t[j+1]=t[j]
7        j=j-1
8      t[j+1]=e
```

Tri à bulle :

```
1  def triBulle(t,n):
2    for i in range(n-1,0,-1) :
3      for j in range(i) :
4        if(t[j]>t[j+1]) :
5          echange(t,j,j+1)
```

Tri Fusion :

```
1  def fusion(t, d, f):
2    r=creerTableau(f-d)
3    m=(d+f)//2
4    i1=d
5    i2=m
6    k=0
7    while (i1<m and i2<f):
8      if (t[i1] < t[i2]):
9        r[k] = t[i1]
10       i1=i1+1
11     else :
12       r[k] = t[i2]
13       i2=i2+1
14     k=k+1
15   while (i1 <m):
16     r[k] = t[i1]
17     i1=i1+1
18     k=k+1
19   while (i2 <f):
20     r[k] = t[i2]
21     i2=i2+1
22     k=k+1
23   for k in range(f-d):
24     t[d+k]=r[k]
```

```
1  def triFusion (t,d,f):
2    if (d<f-1):
3      m=(d+f)//2
4      triFusion(t,d,m)
5      triFusion(t,m,f)
6      fusion(t,d,f)
```

1er Appel :

```
triFusion(t,0,n)
```

Tri rapide :

```
1  def separationPivot(t,n, d, f):
2    p = d
3    i = d
4    j = f
5    while(i<=j):
6      if (t[i]<t[p]):
7        i=i+1
8      else :
9        echanger(t,i,j)
10       if (i==p):
11         p=j
12       else :
13         if (j==p):
14           p=i
15       j=j-1
16   echanger (t, i, p)
17   return (i)
```

```
1  def triRapideRec(t, n, d, f):
2    if(d<f):
3      p=separationPivot(t,n,d,f)
4      triRapideRec(t,n,d,p-1)
5      triRapideRec(t,n,p+1,f)

   def triRapide(t, n):
     triRapideRec(t,n,0,n-1)
```