1ère Spé. NSI Gr. 2
Lycée Fustel de Coulanges

Contrôle #4, 05/04/2024
Durée: 1 Heure

Ce contrôle comporte 5 questions; le nombre maximal possible de points est de 22. Les réponses sont à porter sur une copie comportant votre nom. Il n'est pas nécessaire de répondre aux questions dans l'ordre — commencez par celles où vous vous sentez le plus à l'aise *(mais ne tentez les questions bonus qu'après avoir fini le reste!!)*.
Les calculatrices ne sont pas autorisées.

---

1. *Questions de cours*
   (a) (1 ½ points) QCM – les réponses sont à porter sur votre copie; il n'est pas nécessaire de justifier vos réponses; il y a une bonne réponse par question
      i. Si j'ai déterminé qu'un algorithme était de complexité quadratique (également noté $\mathcal{O}(n^2)$), et si je double la taille des données en entrée, alors la durée de traitement...
         ◯ ... restera la même.
         ◯ ... doublera.
         ◯ ... triplera.
         ◯ ... quadruplera.

```python
def compte_elts(liste):
    compteur = 0
    for element in liste:
        compteur += 1
    return compteur
```

      ii. La fonction ci-dessus a une complexité en (n étant la taille de `liste`)...
         ◯ ... $\mathcal{O}(n^2)$.
         ◯ ... $\mathcal{O}(n)$.
         ◯ ... $\mathcal{O}(log_2(n))$.
         ◯ ... $\mathcal{O}(1)$
      iii. Un tri "stable"...
         ◯ ... maintient les positions relatives des éléments équivalents.
         ◯ ... place les éléments en ordre décroissant.
         ◯ ... a une complexité supérieure à un tri instable.
         ◯ ... a une complexité inférieure à un tri instable.
   (b) (2 points) Expliquez en deux ou trois phrases le principe d'une approche gloutonne à la résolution d'un problème.
   (c) (2 points) Supposons que la fonction `recherche_dicho(table, elt)` implémente la recherche dichotomique telle que nous l'avons étudiée en cours. Quelles sont les valeurs qui vont être examinées lors de l'appel `recherche_dicho([0, 1, 1, 2, 3, 5, 8, 9, 11], 7`? *Indice: la première va être table[4] qui vaut 3.*[1]

2. (6 points) *Détermination de complexités*

```python
def f1(n):
    x = 0
    for i in range(n):
        x = x + 1
    return x
```

```python
def f2(n):
    x = 0
    for i in range(1000000000):
        x = x + 1
    return x
```

```python
def f3(n):
    x = 0
    for i in range(n):
        for j in range(5):
            x = x + 1
    return x
```

```python
def f4(n):
    x = 0
    i = 0
    while i * i < n:
        x = x + 1
        i = i + 1
    return x
```

```python
def f5(n):
    x = 0
    for i in range(n):
        for j in range(n):
            x = x + 1
    return x
```

```python
def f6(n):
    x = 0
    for i in range(n):
        x = x + 1
    for j in range(n):
        x = x + 1
    return x
```

[1] N'hésitez pas à utiliser sur votre copie un tableau donnant les valeurs successives des indices "`debut`", "`fin`", et "`milieu`".

Déterminer la complexité de chacune des fonctions `f1` à `f6` en fonction de n (vous pouvez utiliser la notation $\mathcal{O}()$ ou la terminologie "constante / linéaire / quadratique / autre"). Il est demandé pour chaque fonction une brève phrase de justification de la réponse.

3. (2 points) *Fonction de recherche de 0*

```
1  def compter_zeros(t):
2  ''' Fonction qui compte le nombre de zéros après chaque élément de t'''
3      n = len(t)
4      compte = [0] * n # Rappel: renvoie un tableau de longueur n de 0: [0, 0, ..., 0]
5      for i in range(n):
6          for j in range(i+1, n):
7              if t[j] == 0:
8                  compte[i] += 1
9      return compte
```

Exécutons cette fonction sur un tableau spécifique: `compter_zeros([1, 0, 2, 0])`. Sur votre copie, complétez le tableau suivant avec les valeurs successives des variables – ajoutez autant de lignes qu'il y aura de passages dans la comparaison de la ligne 7 "`if t[j] == 0:`"[2]; puis concluez en indiquant ce que renverra la fonction.

Ligne Code	compte	i	j	t[i]	t[j]
7	[0,0,0,0]	0	1	1	0
7	[1,0,0,0]	0	2	1	2
7	...	...	...	...	...

4. *Fichier CSV* — On considère le code suivant:

```
1  import csv
2  fichier = open('Specialites.csv', 'r', encoding = 'utf-8')
3  table = list(csv.DictReader(fichier))
```

Et soit le fichier `Specialites.csv` contenant les données suivantes:

```
Eleve,Classe,Age,Spe1,Spe2,Spe3
Loubna,1G5,16,Maths,NSI,Physique
Olivier,1G2,17,NSI,Maths,SES
Lenny,1G2,17,LLC-Anglais,SES,NSI
Anju,1G3,16,LLC-Anglais,NSI,SES
Sophie,1G5,15,Maths,NSI,SES
```

(a) (½ point) Qu'est-ce qui va s'afficher à l'exécution du code suivant?

```
1  for i in range(len(table)):
2      print(table[i]['Spe2'])
```

(b) (1 point) Complétez la fonction suivante pour qu'elle fasse ce qui est spécifié.

```
1  def CompteEleves(spe, table):
2      ''' Fonction qui renvoie le nombre d'élèves ayant choisi la spécialité passée en argument'''
3      n = len(table)
4      compte = 0
5      for i in range(n):
6              # A COMPLETER: le "if" qui va tester si l'élève d'indice i a la spécialité "spe"
7              ...
8                  compte += 1
9      return compte
```

(c) (2 points) Rédigez une fonction AgeMoy(classe) qui renvoie l'âge moyen des élèves présents dans la classe passée en argument. Par exemple `AgeMoy('1G5')` renverra la valeur 15.5 (la moyenne de 15 et 16)[3].

[2] Indice: il y en aura 6 en tout, les deux déjà présents dans l'énoncé inclus.

[3] N'hésitez pas à commencer par en rédiger l'algorithme en pseudo-code: des points seront attribués à cela même si le code final est faux ou absent.

5. *Ré-écriture du tri par sélection*

   (a) (2 points) Ecrivez le pseudo-code d'une fonction `Prochain_Min(liste, indice_courant)` qui prend en entrée une liste et un indice et qui renvoie l'indice de la valeur minimale présente dans `liste` entre l'indice `indice_courant` (inclus) et la fin de la liste. Par exemple: `Prochain_Min([10,13,11,12], 0)` renverra 0 (correspondant à la valeur 10) et `Prochain_Min([10,13,11,12], 1)` renverra 2 (correspondant à la valeur 11).

   (b) (1 point) Traduisez le pseudo-code que vous venez de rédiger en fonction codée en Python.

   (c) (2 points) Complétez le code suivant (parties A et B) pour qu'il réalise le tri par sélection d'une table passée en argument tel que nous l'avons vu en cours[4]:

```
def TriSelect(table):
    ''' Fonction qui applique le tri par sélection à table et renvoie la table triée'''
    n = len(table)
    for i in range(n):
        # A COMPLETER - A: appel à votre fonction Prochain_Min
        ...
        # A COMPLETER - B: permutation des valeurs pour placer le ième plus petit élément à l'indice i
        ...
    return table
```

(Question bonus 1): En conservant le modèle que l'on a utilisé dans la question précédente pour le tri par sélection (une fonction principale et une sous-fonction qui cherche l'indice du minimum), écrivez une implémentation de ce même tri mais en ordre décroissant.

(Question bonus 2): En utilisant ce que vous avez fait à la question 6 *et* ce que vous avez fait à la question bonus 1, écrivez une implémentation du tri par sélection qui classe tous les nombres pairs de la liste par ordre croissant à gauche de la liste en sortie, et tous les nombres impairs par ordre décroissant à droite. Par exemple, si on appelle cette fonction `TriSelTordu(table)`, l'appel `TriSelTordu([1, 9, 8, 10, 6, 5, 11, 23, 2])` renverra [2, 6, 8, 10, 23, 11, 9, 5, 1].

(Question bonus 3): Une technique pour repérer le plagiat dans un texte consiste à repérer les enchainements de mots (plutôt que les mots individuels). Écrivez une fonction `Plagiat(txt)` qui prend en entrée une chaîne de caractères et renvoie les 2 enchainements de mots les plus fréquents qu'elle contient. Par exemple si l'on donne à la variable txt la valeur "Le vent souffle fort sur la plaine. Les arbres dans la plaine se courbent sous le vent. Le vent, le vent, toujours le vent.", `Plagiat(txt)` renverrait les enchainements "le vent" (présent cinq fois) et "la plaine" (deux fois)[5].

[4] Un petit rappel, pour gagner du temps dans la permutation: le code "`a , b = b , a`", en Python, met la valeur de a dans b et celle de b dans a.

[5] Rappel: la commande "`lst = txt.split()`" crée une liste `lst` dont les éléments sont les mots de `txt`.