LYCÉE MARIE CURIE **1NSI**
2019–2020

Devoir surveillé n°4 – mathématiques
03/02/2020

Exercice 1 (Vrai/Faux – 1 point)

Pour chacune des affirmations suivantes, dire si elle est vraie ou fausse et justifier.
Une réponse sans justification ne sera pas comptée.

1. Un algorithme contenant une boucle itérative « Pour i allant de 1 à n » dans laquelle est appelée une fonction de coût linéaire (en fonction de n) a un coût au moins quadratique.

Exercice 2 (QCM – 2 points)

Pour chaque question, une seule réponse parmi celles proposées est exacte. Donner la lettre correspondante. Une erreur fait perdre 1/3 de point pour cette exercice. Une absence de répondre ne fait perdre aucun point. La note minimale sur cet exercice est 0.

1. Dans la fonction ci-contre, la variable `n` est un entier naturel.
   La complexité de l'algorithme (en fonction de `n`) est :
   (a) linéaire (b) quadratique
   (c) celle d'une dichotomie (d) autre

```
def f(n):
  for i in range(n):
    for j in range(5):
      print(i/2+j)
```

2. On considère le code ci-contre.
   Le nombre de sommes effectuées est :
   (a) 8 (b) 12 (c) 16 (d) 20

```
  for i in range(4):
    for j in range(2,4):
      s=i+j
    s=s+i
```

Exercice 3 (7 points)

On considère l'algorithme ci-contre, écrit en langage Python.
La variable L est de type list et contient des valeurs de type float (ou int).

1. Démontrer que l'exécution de la fonction somme termine toujours.
2. Démontrer que la propriété : « res = L[0]+...+L[i-1] »
   (somme qui est vide, donc nulle si i-1<0)
   est un invariant de la boucle.
3. Conclure alors que le résultat retourné par la fonction somme est la somme des termes de la liste L.

```
def somme(L):
  res = 0
  i = 0
  while i<len(L):
    res = res + L[i]
    i = i+1
  return res
```

Exercice 4 (5 points)

On considère l'algorithme ci-contre, écrit en langage Python.
On rappelle que :

randint(a,b) renvoie un entier choisi aléatoirement entre a et b (compris).

t.pop() enlève le dernier élément de la liste `t`.

1. Démontrer que `len(t)-i` est un variant de la boucle. Que peut-on en déduire ?
2. Combien de fois s'exécute la boucle « tant que » ?
3. Que renvoie la fonction `f` ? Le démontrer.

```
from random import randint
def f(t):
  r = 0
  i = 0
  while i<len(t):
    r = 2*r+3*r**2
    if randint(0,2) == 0:
      t.pop()
    else:
      i = i+1
  return r
```