Algorithmes

1. Terminaison

Exercice 1 (Vrai/Faux)
Pour chacune des affirmations suivantes, dire si elle est vraie ou fausse. Justifier si possible.

1. En Python, une boucle `for` unique se termine toujours en un nombre fini d'étapes.
2. Pour démontrer la terminaison d'un programme, on peut utiliser un invariant de boucle.

2. Démonstrations de validité (correction et terminaison)

Exercice 2
Est-il vrai qu'une boucle `while` se termine toujours au bout d'un nombre fini d'itérations si l'algorithme est valide ? Justifier.

Exercice 3
On considère le code Python suivant, où `a` et `b` sont des entiers naturels :

```
def f(a,b):
    s,k=a,b
    while k>0:
        s = s+1
        k=k-1
    return s
```

1. Justifier que la valeur finale de `k` ne peut pas être 1.
2. Démontrer que la propriété « `s+k=a+b` » est un invariant de la boucle `while`.
3. Démontrer que la propriété « `k`$\geqslant$`0` » est un invariant de la boucle `while`.
4. Déduire à l'aide des deux questions précédentes que le résultat renvoyé (c'est à dire `s`) est égal à la somme `a+b`.

Exercice 4
On effectue la division euclidienne de a par b où a et b sont des entiers naturels non nuls. On cherche donc q et r entiers naturels, avec $0 \leqslant r < b$ tels que $a = bq + r$. Voici l'algorithme qui réalise cela :

```
q = 0
r = a
while r >= b:
    q = q+1
    r = r-b
return q,r
```

1. Démontrer que la propriété « `a = bq+r` » est un invariant de la boucle.

2. Démontrer que l'algorithme termine, et qu'à la fin de l'exécution, `0⩽r<b`.

Exercice 5
On considère l'algorithme suivant où n est un entier naturel :

```
i←0
f←1
Tant que i < n Faire
    i←i+1
    f←f×i
Fin Tant que
```

1. Justifier que la boucle n'est pas infinie.
2. On suppose que $n \geqslant 0$.
   Démontrer que la propriété « $f = i!$ et $i \leqslant n$ » est un invariant de cette boucle.
   Rappel : le nombre $i! = 1 \times 2 \times \cdots \times i$ est la factorielle de i.
3. Conclure que l'algorithme calcule la factorielle de n.

Exercice 6
On considère l'algorithme suivant où n est un entier naturel :

```
p←1
Pour i allant de 1 à n Faire
    p←p×2
Fin Pour
```

Trouver un invariant de boucle permettant de démontrer que l'algorithme calcule 2^n.

3. Nombres d'opérations

Exercice 7
Déterminer pour chaque code Python suivant, le nombre d'additions effectuées, éventuellement en fonction de `n`.

```
x=0
for i in range(2):
    x = x+i
    for j in range(3):
        x = x+j
```

```
x=0
for i in range(2):
    x = x+i
    for j in range(n):
        x = x+j
```

```
x=0
for i in range(n):
    x = x+i
    for j in range(n):
        x = x+j
```

Exercice 8
Combien de fois la fonction `print` est-elle appelée dans le code suivant ?

```
for i in range(5):
    for j in range(i+1,5):
        print(i+j)
```

4. Coût

Exercice 9 (Vrai/Faux)
Pour chacune des affirmations suivantes, dire si elle est vraie ou fausse. Justifier si possible la réponse.

1. Si un coût est quadratique, c'est qu'il y a au moins deux boucles imbriquées.
2. S'il faut un temps t pour trouver par dichotomie une valeur dans une liste triée de taille n, alors il faudrait un temps $2t$ environ pour une liste de taille $2n$.
3. Pour trouver une valeur dans une liste triée, l'algorithme de dichotomie est toujours plus efficace que l'algorithme par recherche séquentielle.
4. Le nombre d'étape dans la recherche dichotomique d'une valeur dans une liste triée de longueur n est de l'ordre du nombre de chiffres dans l'écriture binaire de n.

Exercice 10 (QCM)
Pour chaque question, une seule réponse est valable. Indiquer laquelle.

1. Quelle affirmation est vraie ?
   (a) Deux algorithmes de coûts identiques effectuent exactement le même nombre d'opérations.
   (b) Le coût d'un algorithme permet d'évaluer un temps d'exécution.
   (c) Plus le coût d'un algorithme est élevé, plus précis est le résultat.
   (d) Une recherche dichotomique a un coût deux fois moins élevé que celui d'une recherche linéaire (puisque l'on divise par 2).
2. On considère le code suivant, où `n` est un entier naturel :

```
x = 0
while x*x<n:
    x = x+1
```

   Quelle affirmation est vraie ?
   (a) Le coût est linéaire en fonction de n.
   (b) Le coût est quadratique en fonction de n.
   (c) L'expression `x*x` est un variant de la boucle.
   (d) La propriété `x*x<n` est un invariant de la boucle.
3. On considère le code Python suivant, où `n` est un entier naturel :

```
x = 1
while x<n:
    x = 2*x
```

   Quelle affirmation est vraie ?
   (a) Le coût est celui d'une recherche dichotomique.
   (b) Le coût est linéaire en fonction de `n`.
   (c) Le coût est quadratique en fonction de `n`.
   (d) Il est impossible de connaître le coût.