Chapitre 1 : correction et terminaison des algorithmes-Exercices

9 février 2023

Exercice 1
On donne ci-dessous un algorithme de multiplication qui prend en entrée deux entiers a et b.

```
1: r ⟵ 0
2: while b>0 do
3:     b ⟵ b − 1
4:     r ⟵ r + a
5: end while
6: return r
```

1. Proposer des préconditions et de postconditions pour cet algorithme.
2. Montrer que, sous les préconditions choisies, le programme termine quels que soient les arguments a et b.
3. Proposer un invariant pour la boucle `while` et montrer que c'est bien un invariant.
4. Montrer que le programme est fortement correct vis-à-vis de la spécification choisie.

Exercice 2
On rappelle ci-dessous l'algorithme de la division euclidienne, qui prend en entrée deux entiers a et b pour calculer leur pgcd.

```
1: while b > 0 do
2:     a, b ⟵ b, a%b
3: end while
4: return a
```

1. L'algorithme est-il partiellement correct ?
2. Proposer des préconditions et postconditions.
3. Montrer que l'algorithme termine en utilisant un variant de boucle.
4. Montrer qu'il est fortement correct.

Exercice 3
On propose ci-dessous un algorithme qui prend en entrée un entier naturel n et doit retourner la somme $\sum_{k=0}^{n} k$.

```
1: s ⟵ 0
2: for i allant de 0 à n do
3:     s ⟵ s + i
4: end for
5: return s
```

1. Proposer des préconditions et postconditions pour l'algorithme.
2. Justifier rapidement que l'algorithme termine et prouver sa correction au moyen d'un invariant de boucle adapté.

Exercice 4
On donne ci-dessous un algorithme de recherche du plus petit élément d'une liste. L'algorithme prend en entrée une liste de nombres entiers ou flottants L.

```
1: m ⟵ L[0]
2: for i allant de 1 à len(L) − 1 do
3:     if L[i] < m then
4:         m = L[i]
5:     end if
6: end for
7: return m
```

1. Proposer des préconditions et postconditions.
2. Justifier rapidement que l'algorithme termine et montrer qu'il est correct en utilisant un invariant de boucle adapté.

Exercice 5
On dit qu'une liste de nombre entier est un palindrome si elle est identique à la liste obtenue en « retournant »la liste initiale, c'est-à-dire en lisant les éléments de la fin au début.

Par exemple, les liste $[1,2,3,3,2,1]$ et $[0,1,2,3,2,1,0]$ sont des palindromes.

On donne ci-dessous un algorithme prenant en entrée une liste L et déterminant si L est un palindrome.

On rappelle que si x est un nombre réel, $\llcorner x \lrcorner$ désigne le plus grand entier p tel que $p \leq x$.

```
1: n ⟵ len(L)
2: p ⟵ ⌞n/2⌟
3: i ⟵ 0
4: while i < p and L[i] = L[n − 1 − i] do
5:     i ⟵ i + 1
6: end while
7: return i = p
```

1. Proposer des précondition et postcondition pour cet algorithme.
2. Montrer que l'algorithme termine.
3. Montrer que l'algorithme est correct en utilisant un invariant de boucle adapté.

Exercice 6
On revient dans cet exercice sur la recherche par dichotomie d'un élément dans un tableau trié

1. On propose dans un premier temps un algorithme prenant en entrée un tableau de nombres entiers T et déterminant si T est trié.

```
1: i ⟵ 0
2: while i < len(T) − 1 and T[i] ≤ T[i + 1] do
3:     i ⟵ i + 1
4: end while
5: return i = len(T) − 1
```

   Proposer des préconditions et postconditions pour cet algorithme. Montrer qu'il termine et qu'il est correct relativement à cette spécification.

2. On propose maintenant une version de l'algorithme de recherche dichotomique d'un entier elt dans un tableau de nombres entiers **triés** T.

   a. Appliquer l'algorithme « à a la main » pour $T = [0,1,3,4]$ et $elt = 2$. Quelles sont les valeurs finales de a et b?

   b. Montrer que l'algorithme termine.

   c. Proposer une spécification et un invariant de boucle et montrer la correction de l'algorithme.

```
1: a, b ⟵ 0, len(T) − 1
2: c ⟵ (a + b)//2
3: while b − a ≥ 0 and T[c] ≠ elt do
4:    if T[c] < elt then
5:       a ⟵ c + 1
6:    else
7:       b ⟵ c − 1
8:    end if
9:    c ⟵ (a + b)//2
10: end while
11: return b ≥ a
```

Exercice 7

On donne ci-dessous une version de l'algorithme de tri bulle, qui prend en entrée une liste T de nombres entiers ou flottants.

```
1: n ⟵ len(T)
2: for i allant de 0 à n − 2 do
3:    for j allant de 0 à n − 2 − i do
4:       if T[i] < T[i + 1] then
5:          Echanger T[i] et T[i + 1]
6:       end if
7:    end for
8: end for
9: return T
```

1. Proposer une postcondition pour l'algorithme.
2. Proposer un invariant pour chacune des deux boucles et vérifier la correction de l'algorithme.