Informatique Théorique 1. Travaux Dirigés : 3

1 Exercice

On considère le programme suivant (a est un entier naturel strictment positif :)

Axiome: $a \in \mathbb{N}_1$

```
b:= a; n := 0;
while b > 1
do  b := b/2 ;  n := n + 1
done
```

1.1

On propose comme invariant la proposition $b \geq 1 \ \wedge \ b2^n \leq a < b2^{n+1}$ Cette proposition est-elle un invariant utile pour montrer que ce programme calcule bien le logarithme entier par défaut (de base 2) de a ?

1.2

Même question avec la proposition $b \geq 1 \ \wedge \ b2^n \leq a < (b+1)2^n$

1.3

Montrer la correction totale de ce programme. On citera les propriétés arithmétiques utilisées.

2 Exercice

On a vu en cours le programme suivant de recherche de x dans un tableau non vide de taille n : :

```
i := 0 ; found := false;
while (i < n ∧  ¬ found)
{Invariant:
i ∈ 0..n ∧
(forall k ∈ 0..n − 1,  a(k) = x  ⟹  i ≤  k)  ∧
found = true  ⟹  i <  n  ∧  a(i) =  x}
do
    if a[i] = x
   then found := true
   else i := i+ 1
   endif
done
```

Ce programme est-il correct pour la recherche de la première occurrence de x ? Comment ne pas refaire toutes les preuves vues en cours ?

3 Exercice

Un utilisateur trouve que la règle du calcul de Hoare pour l'affectation

$$\{P[e/x]\} \text{ x :=e } \{P\}$$

est peu intuitive, et propose de la remplacer par la règle suivante :

$$\{P\} \text{ x :=e } \{P \wedge x = e\}$$

Montrer que ce n'est pas une bonne idée.

4 Exercice

Soient a et b deux entiers naturels, avec $b > 0$. Montrer que le programme suivant calcule bien le reste de la division entière de a par b. Autrement dit, la valeur finale de la variable r doit vérifier la formule

$$0 \leq r < b \ \wedge \ \exists q \in \mathbb{N}, a = bq + r$$

```
r :=  a ;
while b ≤ r
do
 r := r - b
done
```

5 Exercice

On a vu au cours de la première séance de TD comment spécifier la recherche d'une position où se trouve le plus petit élément d'un tableau d'entiers.

Proposer un programme dans le langage utilisé dans ce cours, et prouver la correction (totale) de votre programme.

6 Exercice

On considère la méthode Java suivante :

```
// returns a position of x in the array a (0 based)
// returns -1 if x doesn't occur in a
// We assume that a is sorted (in increasing order)
//
```

```
 static int search(int x, int a[])  {
  int result = -1;
  int lo = 0;
  int hi = a.length - 1;
  int m = hi/2;
  boolean found = false;
  while (!found & x >= a[lo] & x <= a[hi] )
   if (a[m] == x) {
     result = m;
     found = true;
   }
   else if (a[m] > x) {
     hi = m - 1;
     m = (lo + m - 1)/2;
   }
   else {
     lo = m + 1;
     m = (m + 1 + hi)/2;
 }
  return result;
}
```

On veut prouver qu'il est correct pour la recherche de x dans un tableau de nombres entiers a, non vide et trié par ordre croissant.

6.1

Le décorer avec une pré- et une postcondition pour le corps de la méthode `search`.

6.2

Proposer un invariant de boucle pour la correction partielle de `search`. Le confronter avec les *obligations de preuves associées* (*i.e.* les lemmes à prouver).

6.3

Comment est utilisée l'hypothèse que le tableau est trié ?

6.4

Étudier la correction totale de la méthode `search`.