Année 2018/2019 TD Preuve de programmes

LOGIQUE ET PREUVE DE PROGRAMMES
IF107

Filière : Informatique, Année : 1

Web: `http://herbrete.vvv.enseirb-matmeca.fr/IF107`

1 Spécification de programmes

Exercice 1 (La bonne spécification)

On considère une fonction :

```
int find(int * t, int n, int x)
```

qui cherche `x` dans le tableau `t` de taille `n`. La fonction `find` retourne la position i telle que $\texttt{t}[i] = \texttt{x}$ si `x` appartient à `t` et -1 sinon.

1. Pourquoi la spécification suivante ne correspond pas à celle de la fonction `find` (r est la valeur de retour de `find`) ?
   - PRECOND : $(n \in \mathbb{N})$
   - POSTCOND : $(r = -1) \vee (0 \leq r < n \wedge t[r] = x)$
2. Proposer une spécification correcte pour `find`.

♦

Exercice 2 **(Spécification de programmes)**

Donner des pré-conditions et des post-conditions pour les programmes ci-dessous en logique du 1er ordre. On nommera `r` la valeur retournée par ces fonctions.

1. La fonction `max` qui retourne le maximum de `x` et de `y`

```
int max(int x, int y);
```

2. La fonction `fact` calcule la factorielle de `n`

```
int fact(int n);
```

3. La fonction `abs` calcule la valeur absolue de `x`

```
int abs(int x);
```

4. La fonction `gcd` calcule le plus grand commun diviseur à `a` et `b`

```
int gcd(int a, int b);
```

5. La fonction `xchange` échange les valeurs de `x` et `y`.

```
void xchange(int * x, int * y);
```

6. La fonction `sort` qui trie le tableau `t` à `n` éléments (ordre croissant)

```
void sort(int * t, int n);
```

7. La fonction `array_eq` qui indique si les tableaux `t1` et `t2` à `n` éléments sont identiques (convention usuelle : 0 pour faux et tout autre valeur pour vrai)

```
int array_eq(int * t1, int * t2, int n);
```

♦

2 Preuves de programmes

Dans tous les exercices suivants, on demande de donner une spécification de l'algorithme, puis d'en démontrer la terminaison (avec la méthodes des ensembles bien fondés) et la correction (par induction pour les programmes récursifs, et en prouvant en invariant pour les programmes itératifs).

Exercice 3 (Factorielle)

Les deux algorithmes ci-dessous calculent la factorielle de `n`.

```
1 fact(n):
2   i := 1; f := 1;
3   while (i ≤ n) do
4     f := f*i;
5     i := i+1
6   end;
7   return f
```

```
1 fact_rec(n):
2   if (n = 0) then
3     return 1;
4   else
5     return n*fact_rec(n−1)
```

♦

Exercice 4 (Recherche linéaire)

Les algorithmes suivants recherchent `x` dans le tableau `t` de taille `n`. Ils retournent $true$ si `x` appartient à `t` et $false$ sinon.

```
1 find(t,n,x):
2   i := 0;
3   while ((i < n) ∧ (t[i] ≠ x)) do
4     i := i+1
5   end;
6   return (i < n)
```

```
1 find_rec(t,n,x):
2   if (n = 0) then
3     return false
4   else if (t[n − 1] = x) then
5     return true
6   else
7     return find_rec(t,n−1,x)
8   end
```

♦

Exercice 5 **(Recherche dichotomique)**

Les fonctions suivantes recherchent `x` dans le tableau `t` de taille `n` trié par ordre croissant. Elles retournent $true$ si `x` appartient à `t` et $false$ sinon. Comparer l'invariant à celui de l'exercice précédent.

```
1  binfind(t,n,x):
2    l := 0; r := n−1;
3    p := (l + r) / 2;
4    while ((l ≤ r) ∧ (t[p] ≠ x)) do
5      if (t[p] < x) then
6        l := p + 1
7      else // (t[p] > x)
8        r := p − 1
9      end;
10     p := (l + r) / 2
11   end;
12   return (l ≤ r);
```

♦

```
1  binfind_rec(t,l,r,x):
2    if (l > r) then
3      return false
4    else
5      p := (l + r) / 2;
6      if (t[p] = x) then
7        return true;
8      else if (t[p] < x) then
9        return binfind_rec(t,p+1,r,x)
10     else
11       return binfind_rec(r,l,p−1,x);
12     end
13   end
14
15 binfind(t,n,x):
16   return binfind_rec(t,0,n−1,x)
```

Exercice 6 (Tri par extraction)

1. Soit l'algorithme suivant :

```
1 selection(t, N, m):
2   imax := 0; i := 1;
3   while (i ≤ m) do
4     if (t[i] > t[imax])
5       imax := i
6     end;
7     i := i + 1
8   end;
9   return imax
```

Cet algorithme retourne un indice $imax \in [0; m]$, avec $m < N$, tel que $t[imax]$ est un maximum de t.

(a) Écrivez en logique des prédicats, une pré-condition et une post-condition pour l'algorithme `selection`.

(b) Justifiez la terminaison de `selection` par la méthode des ensembles bien fondés.

(c) Donnez un invariant de boucle et justifiez la correction de `selection`.

On considère une fonction $\texttt{swap}(t, i, j, N)$ qui retourne un tableau t' identique à t sauf les valeurs d'indices i et j qui ont été permutées. Sa spécification est la suivante :

PRECOND : $(0 \leq i < N) \wedge (0 \leq j < N)$

POSTCOND : $(t'[i] = t[j]) \wedge (t'[j] = t[i]) \wedge (\forall k.\, (k \neq i \wedge k \neq j) \implies (t'[k] = t[k])$

2. Soit l'algorithme `sort` suivant :

```
1 sort(t, N):
2   i := N − 1; imax := 0;
3   while (i ≥ 1) do
4     imax := selection(t, N, i);
5     t := swap(t, N, i, imax);
6     i := i − 1
7   end
```

La fonction $\texttt{sort}(t, N)$ trie le tableau t à N éléments par la méthode d'extraction du maximum.

(a) Écrivez en logique des prédicats, une pré-condition et une post-condition pour l'algorithme `sort`.

(b) Justifiez la terminaison de `sort` par la méthode des ensembles bien fondés.

(c) Donnez un invariant de boucle et justifiez la correction de `sort`.

♦