TD5 – preuve de programmes

Exercice 1

Soient les triplets de Hoare suivants :

(a) {z == y + 1} x = z * 2 ; {x == 4}

(b) {y == 7} x = y + 3 ; {x > 5}

(c) {false} x = 2 / y ; {true}

(d) {y < 16} x = 2 / y ; {x < 8}

1. Quels sont les triplets de Hoare invalides ?
2. Pour chaque triplet de Hoare valide, déterminez la plus forte post-condition par rapport à la pré-condition donnée.
3. Pour chaque triplet de Hoare valide, déterminez la plus faible pré-condition par rapport à la post-condition donnée.

Exercice 2

1. Calculez la plus faible pré-condition P pour chacun des triplets de Hoare suivants :
   (a) {P} x = y * 2 ; {x == y * 2}
   (b) {P} x = x + 3 ; {x == z}
   (c) {P} x = x + 1 ; y = y * x ; {y == 2 * z}
   (d) {P} x = 0 ; {x == 1}
   (e) {P} x = 0 ; {true}
   (f) {P} `if` (x > 0) y = x ; `else` y = 0 ; {y > 0}

Exercice 3

Soit le triplet de Hoare suivant :

```
{N >= 0}
i = 0;
while (i < N)
  i = N;
{i == N}
```

1. Parmi les conditions suivantes, lesquelles sont des invariants de boucle permettant de prouver la post-condition ? Pour celles qui ne sont pas correctes, expliquez pourquoi.
   (a) i == 0
   (b) i == N
   (c) N >= 0
   (d) i <= N

Exercice 4

1. Soit la fonction suivante :

```
public int square(int n) {  // n >= 0
  int cpt = n;
  int r = 0;
  while (cpt > 0) {
    r = r + n;
    cpt = cpt - 1;
  }
  return r;
}
```

   (a) Que calcule cette fonction ?

   (b) Écrivez le triplet de Hoare correspondant à cette fonction.

   (c) Déterminez l'invariant de boucle.

   (d) Démontrez la validité du triplet en appliquant les règles de déduction de la logique de Hoare.

2. Soit la fonction suivante :

```
public int isqrt(int n) { // n >= 0
  int r = 0;
  while (n >= (r+1)*(r+1)) {
    r = r+1;
  }
  return r;
}
```

   (a) Mêmes questions que précédemment.

3. Soit la fonction suivante :

```
public int fact(int n) { // n >= 0
  int t = n;
  int r = 1;
  while (t != 0) {
    r = r * t;
    t = t - 1;
  }
  return r;
}
```

   (a) Mêmes questions que précédemment.

4. Soit la fonction suivante :

```
public int sum(int n, int[] t) { // n >= 0
  int j = 0;
  int s = 0;
  while (j < n) {
    s = s + t[j];
    j = j + 1;
```

```
  }
  return s;
}
```

(a) Mêmes questions que précédemment.

5. Soit la fonction suivante :

```
public int pow(int a, int n) { // a >= 0 et n >= 0
  int b = a;
  int m = n;
  int r = 1;
  while (m > 0) {
    if (m % 2 == 0) {
      b = b * b;
      m = m / 2; // division entiere
    } else {
      r = r * b;
      m = m - 1;
    }
  }
  return r;
}
```

(a) Mêmes questions que précédemment.

6. Soit la fonction suivante :

```
public int mult(int x, int y) { // x >= y >= 0
  int r = 0;
  int a = x;
  int b = y;
  while (b > 0) {
    if (b % 2 == 1) {
      r = r + a;
    }
    a = 2 * a;
    b = b / 2; // division entiere
  }
  return r;
}
```

(a) Mêmes questions que précédemment.