IHDCB331 - Algorithmique II
Devoir 3

Cédric Evrard

October 27, 2019

1 Question 1

1. Tous les éléments de a[i], a[i + 1], ... a[j] sont égaux à zéro

```
/*@ invariant (\forall int i;
 @       i >= 0 && i < j;
 @       a[i] == 0);
 @*/
```

2. Tous les éléments de a sont distincts

```
/*@ invariant (\forall int i;
 @       i >= 0 && i < arrayDistinct.length;
 @       (\forall int j; j >= 0 && j < a && i != j; a[j] != a[i]));
 @*/
```

3. Tous les nombres de a sont pairs

```
/*@ invariant (\forall int i;
 @       i >= 0 && i < a;
 @       a[i] % 2 == 0);
 @*/
```

4. Tous les nombres de a sont inférieurs ou égaux à 2

```
/*@ invariant (\forall int i;
 @       i >= 0 && i < a.length;
 @       a[i] <= 2);
 @*/
```

5. Tous les nombres pairs de a[i], a[i + 1], ... a[j] sont inférieurs à 10

```
/*@ invariant (\forall int i;
  @       i >= 0 && i < j;
  @       (a[i] % 2 == 0 && a[i] < 10) || a[i] % 2 != 0);
  @*/
```

6. Il existe une valeur zéro dans a[i], a[i + 1], ... a[j]

```
/*@ invariant (\exists int i;
 @       i >= 0 && i < j;
 @       a[i] == 0);
 @*/
```

7. Les éléments de a sont triés par ordre croissant

    ```
    /*@ invariant (\forall int i;
     @       i >= 0 && i < a - 1;
     @       a[i] <= a[i + 1]);
     @*/
    ```

8. x est le minium de a

   ```
   /*@ normal_behavior
    @ assignale \nothing;
    @ ensures \forall (int i; i >= 0 && i < a.length; a[i] >= \result);
    @ ensures \exists (int i; i >= 0 && i < a.length; a[i] == \result);
    @*/
   ```

2 Question 2

1. **double** max(**double** x, **double** y)

   ```
   /*@ normal_behavior
    @ assignable \nothing;
    @ ensures (a > b && \result == a) || (a <= b && \result == b);
    @*/
   ```

2. **boolean** contient(**double** [] a, **double** x)

   ```
   /*@ normal_behavior
    @ assignable \nothing;
    @ ensures \result == (\exists int i;
    @      i >= 0 && i < a.length;
    @      x == a[i]);
    @*/
   ```

3. **double** max(**double** [] a)

   ```
   /*@ normal_behavior
    @ assignale \nothing;
    @ ensures \forall (int i; i >= 0 && i < a.length; a[i] <= \result);
    @ ensures \exists (int i; i >= 0 && i < a.length; a[i] == \result);
    @*/
   ```

4. **int** factorielle(**int** n)

   ```
   /*@ normal_behavior
    @ assignable \nothing;
    @ requires n >= 1;
    @ ensures \result == (\product int i; i >= 1 && i <= n; i);
    @*/
   ```

5. **int** intSqrt(**int** n)

   ```
   /*@ normal_behavior
    @ assignable \nothing;
    @ requires n >= 0;
    @ ensures \result * \result <= n;
    @ ensures (\result + 1) * (\result + 1) > n;
    @ ensures \result >= 0
    @*/
   ```

3 Question 10

La question 10 se base sur la signature de la méthode suivante : *public boolean enLigne(int[] a, int n, int x)* où le tableau de **String** à été remplacé par un tableau de **int**

```
/*@ normal_behavior
 @ assignable \nothing;
 @ ensures \result == (\exists int i; i >= 0 && i < a.length - n;
 @      (\forall int j; j >= i && j <= i + n; a[j] == x));
 @*/
```

4 Question 12

```
/*@ normal_behavior
 @ assignable \nothing;
 @ ensures \result == (\sum int i; i >= 0 && i < l; a[i]);
 @*/
public boolean int somme(int[] a, int l) {
    int somme = 0;
    int i = 0;

    /*@ loop_invariant i >= 0 && i < l;
     @ loop_invariant somme == (\sum int j; i >= 0 && j < i; a[j]);
     @ decreases l - i;
     @*/
    while (i < l) {
        somme = somme + a[i];
        i = i + 1;
    }

    return somme;
}
```

5 Question 13

```
/*@ normal_behavior
 @ assignable \nothing;
 @ ensures \result == m + n
 @*/
public int somme (int m, int n) {
    int res = m;
    int i = 0;

    /*@ loop_invariant i >= 0 && i < n;
     @ loop_invariant res == \old(res) + 1 && i = \old(i) + 1;
     @ decreases n - i
     @*/
    while (i < n) {
        res = res + 1;
        i = i + 1;
    }
    return res; // Correction par rapport a l'enonce ou une variable result
       qui n'existe pas est retournee
}
```

6 Question 15

```
/*@ normal_behavior
 @ assignable nothing;
 @ requires true;
 @ ensures (a > b && \result == a) || (a <= b && \result == b)
 @*/
int max(int a, int b) {
    if (a > b) {
        return a;
    } else {
        return b;
    }
}
```

7 Question 22

```
public class Tri {

    /*@ normal_behavior
     @ assignable a;
     @ ensures \forall int i; i >= 0 && i < a.length - 1; a[i] < a[i + 1]
     @*/
    static void tri(int[] a) {
        boolean tri;

        /*@ loop_invariant trie == true && \forall int j; j >= 0 && j <
           a.length - 1; a[j] < a[j + 1]
         @ loop_invariant trie == false && \exists int j; j >= 0 && j <
           a.length - 1; a[j] > a[j + 1]
         @*/
        do {
            trie = true;

            /*@ loop_invariant i >= 0 && i <= a.length - 1;
             @ loop_invariant trie == true && \forall int j; j >= 0 && j < i
           - 1; a[j] < a[j + 1]
             @ loop_invariant trie == false && \exists int j; j >= 0 && j < i
           - 1; a[j] > a[j + 1]
             @*/
            for (int i = 0; i < a.length - 1; i++) {
                if (a[i] > a[i + 1]) {
                    /* ... */
                }
            }
        } while (!trie)
    }
}
```