

**TD : Spécifications et Preuves Formelles**

---

**Exercice 1** Logique de Hoare.

Pour chaque suite d'instructions ou programme donné, montrer qu'elle satisfait sa spécification en utilisant les règles de la logique de Hoare.

## 1. Spécification :

Précondition :  $i = i_0, j = j_0$ Postcondition :  $i = j_0, j = i_0$  $i = i - j;$  $j = i + j;$  $i = j - i;$ 2. Précondition :  $n = n_0, m = m_0$ Postcondition :  $res = n_0 * m_0$  $res = 0;$  $if (n < 0) \{$  $n = -n;$  $m = -m;$  $\} else \{$  $\}$  $while (n != 0) \{$  $res = res + m;$  $n = n - 1;$  $\}$ 3. Précondition :  $n = n_0 \wedge n_0 \geq 0$ Postcondition :  $res = n_0!$  $res=1;$  $while (n != 0) \{$  $res = res * n;$  $n = n - 1;$  $\}$ **Exercice 2** On donne le programme  $I$  $res=1; j=n;$  $while (j>1) \{$  $res = res * j;$  $j = j - 1;$  $\}$

1. Montrer que  $res^*j != n!$  est un invariant de la boucle while.
2. Est-ce que cette invariant permet de prouver  $\{n \geq 0\}I\{res = n!\}$ ? Si ce n'est pas le cas proposer un autre invariant permettant de prouver la spécification.

**Exercice 3** Plus faible précondition, plus forte post-condition.

$I$  est un programme qui utilise les variables  $x_1, \dots, x_n$ . Tous les prédicats sont des formules dont les variables libres sont incluses dans  $x_1, \dots, x_n$ . Un modèle d'un prédicat  $P$  est une affectation de valeur aux variables qui rend vraie la formule  $P$ . Un prédicat  $P$  est plus faible qu'un prédicat  $P'$ , noté  $P' \succ P$ , si  $P' \Rightarrow P$  est vraie (tout modèle de  $P'$  est aussi un modèle de  $P$ ).

1. Déterminer quel est le prédicat le plus faible? même question avec le prédicat le plus fort.
2. Est-ce que  $(x > 2 \wedge y > 0) \Rightarrow x > 1$ ? Est-il vrai que  $\{(x > 2 \wedge y > 0)x = y + 1; \{x > 1\}$ ?
3. Etant donné un programme  $I$  et un prédicat  $Q$ , la plus faible précondition de  $I$  pour  $Q$  est le prédicat  $wp(I, Q)$  tel que  $\{wp(I, Q)\}I\{Q\}$  et pour tout  $P$  tel que  $\{P\}I\{Q\}$  alors  $P$  est plus fort que  $wp(I, Q)$  (c.a.d.  $P \succ wp(I, Q)$ ). Montrer que  $wp(x = y + 1; , x > 1)$  est  $y > 0$ .
4. Proposer une définition de plus forte postcondition relative à un prédicat  $P$  et à un programme  $I$ .

**Exercice 4** On considère un tableau d'entiers  $a$  indicé de 0 à  $n - 1$ . Ecrire les expressions JML pour spécifier :

1. Que la partie du tableau entre  $j$  et  $k$  ne contient que des éléments avec la même valeur. La spécification doit exprimer que  $j$  et  $k$  sont des indices valables et que  $j$  n'est pas plus grand que  $k$ .
2. Que la partie de tableau entre  $j$  et  $k$  (inclus) contient un élément égal à 0. Même question avec *un et un seul élément égal à 0*.
3. Que la partie de tableau entre  $j$  et  $k$  (inclus) ne contient aucun élément égal à 0.
4. Spécifier la méthode `int [] inverse (int [] a)` qui renvoie un tableau  $res$  qui inverse l'ordre des éléments de  $a$ . Ici spécifier consiste à donner les préconditions et postconditions de la méthode.
5. Même question mais la méthode a le profil `void inverse (int [] a)` et inverse  $a$  sur lui-même.
6. Spécifier la méthode `void simplifie(int [] a)` qui élimine toutes les duplications d'éléments de  $a$ .

**Exercice 5** Drapeau tricolore (hollandais à l'origine).

Le problème est d'ordonner une suite d'éléments colorés par 3 couleurs différentes *bleu*, *blanc*, *rouge* de manière à obtenir un drapeau français. Cela est équivalent à ordonner un tableau dont les éléments sont 1 (code de bleu), 2 (code de blanc), 3 (code de rouge) de manière à mettre tous les 1 en tête, suivis de tous les 2, suivis de tous les 3.

1. Ecrire une spécification de la méthode qui prend en argument un tableau d'entier et renvoie le tableau trié correspondant à l'ordre du drapeau français, une exception est renvoyé en cas de couleur autre.
2. Définir un jeu de test à partir de la spécification.
3. Trouver un invariant pour l'opération d'échange et en déduire un programme qui résoud le problème du drapeau tricolore.
4. Donner le résultat du programme sur le jeu de test défini à la question 2.

**Exercice 6** Racine carrée.

On veut écrire un programme *int racine(int n)* permettant de calculer la racine carrée entière d'un entier positif ou nul (on approche la racine carrée réelle par l'entier immédiatement inférieur).

1. Ecrire une spécification de la méthode *racine*.
2. Définir un jeu de test à partir de la spécification.
3. On donne le programme *int racine(int n)* dont le corps est :

```
res := 0;
somme := 1;
while somme <= n do
  res := res + 1;
  somme := somme + 2 * res + 1
```

Satisfait-il votre jeu de test ?

4. Prouver dans la logique de Hoare que ce programme correspond à votre spécification (le résultat retourné est *res*).