Université de Strasbourg
Certification du logiciel Feuille de TP

TP noté : preuves de programmes avec Frama-C

I) Modalités

Pour utiliser frama-c, il faut se connecter à la machine `axis.u-strasbg.fr` via la commande suivante `ssh -Y username@axis.u-strasbg.fr`. Un nom d'utilisateur personnel `username` ainsi que le mot de passe associé vous sera remis au début du contrôle. Pour lancer frama-c avec le plugin jessie, utiliser la commande suivante : `frama-c -jessie moncode.c`

Attention, il n'est pas utile de créer une fonction `main`, on écrira uniquement les fonctions demandées sans utiliser de `#include` et dans des fichiers séparés.

Tous les exercices seront à rendre en un seul fichier zip sous moodle.

II) Eléments de syntaxe

1. Syntaxe des annotations :

```
formula ::= expr
          | expr rel expr
          | formule ==> formule
          | formule <==> formule
          | formule && formule
          | formule || formule
          | \forall type ident ; formule
          | \exists type ident ; formule
rel ::=  == | != | < | <= | > | >=
```

2. Syntaxe de pré et post-conditions :

```
/*@ requires ...;
    ensures ...; */
```

3. Annotation des boucles :

```
/*@
   loop invariant ...;
   loop variant ...;
*/
```

4. Le prédicat `\valid(x)` spécifie que x est bien une zone mémoire valide.
5. Le prédicat `\valid(t+(0..n-1))` spécifie que les zones mémoire `t[i]` pour `i` dans `[0... n-1]` sont valides.
6. On désigne le résultat de la fonction par `\result`. On désigne l'ancienne valeur d'une variable x par `\old(x)`.
7. La commande `assigns` permet de spécifier quels sont les variables modifiées. Exemple `assings t[0..n-1]` permet de spécifier que le tableau t est modifié. La commande `assigns \nothing` permet de spécifier que la fonction ne réalise pas d'effet de bord.

III) Exercice 1 (à rendre sur moodle.unistra.fr)

1. Définir une fonction `min2` : `int min2(int a, int b)` qui retourne le minimum des deux arguments `a` et `b`. Spécifier cette fonction et utiliser frama-c pour vérifier que votre programme respecte bien sa spécification.
2. Etendre cette fonction pour qu'elle accepte trois arguments `min3` : `int min3(int a, int b, int c)`. Adapter également la spécification.
3. Ecrire maintenant une fonction `min4` : `int min4(int a, int b, int c,int d)` calculant le minimum de ces 4 valeurs en utilisant uniquement la fonction `min2`. Spécifier cette fonction et utiliser frama-c pour vérifier que votre programme respecte bien sa spécification.

IV) Exercice 2 (à rendre sur moodle.unistra.fr)

1. Ecrire une fonction `int up_and_down(int t[], int taille, int u)` qui prend en argument un tableau `t`, sa taille `taille` et une position `u` dans le tableau. Cette fonction vérifiera que le tableau est bien trié par ordre croissant jusqu'à la case `u`, puis que le tableau est trié par ordre décroissant de la case `u` jusqu'à la fin. On utilisera de préférence une boucle `for` et on choisira d'exécuter l'instruction `return 0` dès que la condition ne sera pas vérifiée.
2. Spécifier cette fonction dans le langage ACSL de spécification de frama-c. On pourra distinguer deux cas dans l'invariant de boucle selon que l'on se trouve avant ou après l'élément d'indice `u` du tableau.
3. A l'aide de frama-c, prouver que la fonction vérifie bien ses spécifications ainsi que sa terminaison.

V) Exercice 3 (à rendre sur moodle.unistra.fr)

1. Programmer une fonction en langage `C` qui prend en argument un tableau t et sa taille n, et qui retourne :
   - 1 si tous les éléments d'indice pair du tableau sont pairs et si tous les éléments d'indice impair du tableau sont impairs.
   - 0 sinon.
2. Spécifier cette fonction dans le langage ACSL de spécification de frama-c.
3. A l'aide de frama-c, prouver que la fonction vérifie bien ses spécifications ainsi que sa terminaison.