

TD – WP

Rendu

Le but est de rendre un rapport présentant vos réponses aux questions de ce TD. Vous devez me rendre le rapport et **uniquement** le rapport. Il doit donc présenter **tous les éléments nécessaires** pour démontrer que vous avez bien compris la question et bien répondu à la question (et donc être illustré par des screenshots des éléments importants, par des copier/coller des morceaux de code importants, etc). **Mettez l'accent sur vos explications**, ce sont elles qui importent le plus. Le résultat n'a de valeur qu'en lien avec son côté explicable (comme lors d'un audit réel en fait).

Le rapport doit être dans l'un de ces formats :

- PDF (préféablement)
- DOCX (et surtout PAS DOC)
- ODT

Format du nom de fichier : td-wp-Nom1-Nom2.pdf

Envoyez votre rapport à l'adresse allan.blanchard@cea.fr au plus tard le 21/03 minuit avec comme objet :

[INSA CVL] WP

Information importante

Si vous n'utilisez pas la VM toutes les commandes `frama-c-gui` sont à remplacer par `ivette`.

1 Diviseur

La fonction suivante renvoie vrai si la valeur `a` est bien divisible par `b`.

```
//@ ensures \result == 1 ==> a % b == 0 ;
int divides(int a, int b){
    if(a % b == 0) return 1;
    else return 0;
}
```

1. Lancez la commande :

```
frama-c-gui divides.c -wp
```

La spécification est-elle vérifiée ? Que pouvons-nous dire de ce couple fonction/contrat ?

2. Lancez la commande :

```
frama-c-gui divides.c -wp -wp-rte
```

Expliquez le problème relevé par WP. Que pouvons-nous faire pour le régler ?

3. Pour tester la qualité de notre spécification, ajoutons le programme suivant :

```
void test_1 (void) {
    int a = 6 ;
    int b = 3 ;
    int result = divides(a, b);
    //@ assert result == 1 ;
}
```

L'assertion est-elle prouvée ? **Pourquoi ?** Compléter la spécification de la fonction `divides` pour assurer que ce soit le cas et **expliquez** votre changement.

4. Pour tester notre spécification, ajoutons le programme suivant :

```
void test_2 (void) {
    int a = 7 ;
    int b = 3 ;
    int result = divides(a, b);
    //@ assert result == 0 ;
}
```

L'assertion est-elle prouvée ? Si ce n'est pas le cas, **expliquez** pourquoi et complétez la spécification de la fonction `divides` pour assurer que ce soit le cas, `test_1` doit bien entendu toujours être prouvée, **expliquez** le changement réalisé.

5. Ajoutons cette déclaration en tête de fichier :

```
extern int h ;
```

Et ajoutons ce test :

```
void test_3 (void) {
    int a = 7 ;
    int b = 3 ;
    h = 21 ;
    int result = divides(a, b);
    //@ assert h == 21 ;
}
```

L'assertion est-elle prouvée ? Si ce n'est pas le cas, **expliquez** pourquoi, complétez la spécification de la fonction `divides` pour assurer que ce soit le cas **et expliquez le changement**, les deux tests précédents doivent continuer à réussir.

2 Échange

À partir de cet exercice, on utilise toujours `-wp-rte`.

La fonction suivante échange deux valeurs :

```
void swap(int* a, int *b) {
    int tmp = *a ;
    *a = *b ;
    *b = tmp ;
}
```

1. Écrivez une clause `ensures` qui exprime l'échange effectué par cette fonction. Nous rappelons que la fonction builtin `\old(...)` permet d'obtenir la valeur d'une expression telle qu'elle était en précondition. Vérifiez que la postcondition est prouvée à l'aide de la commande :

```
frama-c-gui -wp swap.c
```

2. Lancez la commande suivante :

```
frama-c-gui -wp -wp-rte swap.c
```

Toutes les propriétés sont-elles prouvées? Et pourquoi? Si tout n'est pas prouvé, ajoutez une pré-condition requises à cette fonction pour permettre la preuve et expliquez en quoi ce changement est correct.

3. Ajoutez une déclaration :

```
extern int h ;
```

Avant votre fonction et une fonction de test :

```
int main(void) {
    h = 42 ;
    int a = 1 ;
    int b = 2 ;
    swap(&a, &b);
    //@ assert h == 42 ;
}
```

4. L'assertion est-elle prouvée? Pourquoi? Si ce n'est pas prouvé, ajoutez une clause assigns à la spécification de swap pour assurer que cette assertion soit prouvée.

3 Pointeurs

La fonction suivante calcule le quotient et le reste de la division de deux entiers :

```
void div_rem(unsigned *x, unsigned *y, unsigned* q, unsigned* r) {
    *q = *x / *y ;
    *r = *x % *y ;
}
```

Spécifiez le programme précédent. En testant la spécification au fur et à mesure. En particulier :

- décrivez le but de chaque test ajouté,
- expliquez en quoi la spécification fixe le test en question.

4 Nombre premier

La fonction suivante renvoie vrai si et seulement si le nombre donné en entrée est premier.

```
int is_prime(unsigned x) {
    if(x <= 1) return 0;
    if(x == 2) return 1;
    if(x % 2 == 0) return 0;

    for(unsigned d = 3 ; d < x ; d++){
        if(x % d == 0) return 0;
    }
    return 1;
}
```

Donnez un contrat cette fonction puis complétez les annotations pour que la commande :

```
frama-c-gui prime.c -warn-unsigned-overflow -wp -wp-rte
```

indique une preuve réussie. **Expliquez** le rôle de chaque annotation.

5 Égalité de vecteurs

La fonction suivante renvoie vrai si et seulement si les deux vecteurs transmis en entrée de fonction sont égaux :

```
int equal(const int* a_1, const int* a_2, size_t len){
    for(size_t i = 0 ; i < len ; ++i){
        if(a_1[i] != a_2[i]) return 0 ;
    }
    return 1 ;
}
```

1. Produisez une spécification pour cette fonction, annotez la boucle avec les bons invariants et prouvez sa correction totale.
2. Écrivez une spécification et un corps pour la fonction suivante et prouvez sa correction. Aucune boucle n'est nécessaire.

```
int different(const int* a_1, const int* a_2, size_t len);
```

6 Chercher et remplacer

Implémentez, spécifiez et prouvez la correction d'une fonction "rechercher et remplacer" de prototype suivant :

```
void replace(int* array, size_t length, int old, int new);
```

Expliquez comment vous avez élaboré les tests de vos spécifications.