Hardware Software Interface

TP2 : Nombres entiers et flottants

Alexis DALEY

Question 1. Voici une fonction pour calculer la somme des éléments d'un tableau :

```
/* WARNING: this function is buggy */
float sum_elements (float a[], unsigned length) {
   int i;
   float result = 0;
   for (i = 0; i <= length - 1; i++)

      result += a[i];
   return result;
}
```

Trouvez l'erreur et corrigez-la !

Question 2. Voici une fonction pour vérifier si une chaîne est plus longue que l'autre :

```
/* WARNING: this function is buggy */
#include <string.h>
int strlonger (char* s, char* t) {
   return strlen(s) - strlen(t) > 0;
}
```

Trouvez l'erreur et corrigez-la !

Question 3. Vrai ou faux, en supposant int 32 bits ?

```
int x = foo();
int y = bar();
unsigned ux = x;
unsigned uy = y;
```

1) (x > 0) || (x-1 < 0)
2) (x & 7) != 7 || (x << 29 < 0)
3) (x * x) >= 0
4) x < 0 || -x <= 0
5) x > 0 || -x >=0
6) x + y == uy + ux
7) x * ~y + uy * ux == -x

Nous utilisons les mêmes règles que le TP précédent :
Vous allez programmer avec les contraintes suivantes :

- *il est interdit d'utiliser* :
    - conditionnel (if ou ?:), boucle, switch, appel de fonction, macro
    - division, modulo et multiplication
    - comparaisons relatives (<, >, <= et >=)
- il est possible d'utiliser :
    - opérations logiques et bit-wise
    - décalage <<, >>
    - addition et soustraction
    - test d'égalité (==, !=)
    - INT_MIN et INT_MAX
    - casting

Question 4. Écrire la fonction int fits_bits(int x, int n) qui retourne 1 si x peut être représenté en n bits en complément à 2 et retourne 0 sinon. On suppose que : 1 <=n <= 32.
Ex :
fits_bits(5,3)=0
fits_bits(-4,3)=1

Question 5. Écrire la fonction int odd_ones(unsigned x) qui retourne 1 quand x contient un nombre impair d'1 et retourne 0 sinon. Le code doit contenir au maximum 12 opérations arithmétiques, logiques et bitwise.