15-122: Principles of Imperative Computation

QuickCheck 2 Solutions

Today, you can hold on to your copy of this QuickCheck and your TA will go over the answers at the end. The main purpose today is to give you some practice before the graded quiz (which should be available Sunday night - keep an eye on Piazza for details). We hope you will use the feedback from this QuickCheck to improve on your weaker areas before the quiz.

Name:

Andrew ID:

Section (circle one): A B C D E F G H

Simplicio and *Sagredo* (you) are a couple of 122 students. They are trying to complete an exercise, but are a bit stuck. Help them out!

Simplicio does not believe that `for` and `while` loops can be used interchangeably. For example, he believes that the following code cannot be written with a `while` loop, while maintaining the loop invariants.

```
1 for(int i = 0; i < n-2; i++)
2 //@loop_invariant i >= 0;
3 {
4    A[i+2] = A[i+1] + A[i];
5 }
```

Show him that you can do it!

Solution:

```
1 int i = 0;
2 while(i < n-2)
3 //@loop_invariant i >= 0;
4 {
5    A[i+2] = A[i+1] + A[i];
6    i++;
7 }
```

Simplicio now thinks that you can convert `for` loops to `while` loops but not vice-versa. Write the following `while` loop as a `for` loop to show him that he's wrong!

```
1 int e = y;
2 while (e > 0)
3 //@loop_invariant e >= 0;
4 //@loop_invariant POW(x,y) == accum * POW(b,e);
5 {
6    accum = accum * x;
7    e = e - 1;
8 }
```

Go *Sagredo*!

Solution:

```
1 for(int e = y; e > 0; e--)
2 //@loop_invariant e >= 0;
3 //@loop_invariant POW(x, y) == accum * POW(b, e);
4 {
5   accum = accum * x;
6 }
```

Simplicio concedes defeat and you both move on to finish the rest of the exercise.

And he's stuck again - this time he's having a problem with the two's complement representation. If $n =$ `0x68A1`, calculate $-n$ (assume that the standard 32 bit representation of ints in C0 is followed)

Solution:

```
n = 0x000068A1, ~n = 0xFFFF975E
-n = ~n + 1 = 0xFFFF975F
```

Also, *Simplicio* has been given a color, `color = 0x18962213`. He needs the color with the same RGB composition, but which is completely opaque.

Solution:

`0xFF962213`

BONUS (if you're done early, try these out!)

Seeing that you're basically smarter that poor *Simplicio* is, *Salviati*, the TA, decides to give you something more challenging.

He has a magic color represented as `int magic`. He won't tell you what `magic` is, but wants you to come up with a way to store an opaque version of `magic` in `magical` *(Hint: Try to use bitwise operators)*

Solution:
int magical = magic | (0xFF « 24);

OR

pixel magical = magic | (0xFF « 24);

Salviati now wants you to do an in place swap of two numbers x and y, without using a temporary variable to store one value while modifying the other. Complete the code below to achieve this.

```
1 void inPlaceSwap (int x, int y){
2   x = x^y;
3   y = y^x;
4   x = x^y;
5 }
```

Solution:

```
x ^ y
because,
1) x' = x ^ y; y = y
2) x' = x ^ y; y' = y ^ (x ^ y) = x
3) x'' = x' ^ y' = (x ^ y) ^ y = x; y' = x
```