15-122 : Principles of Imperative Computation, Fall 2013

Written Homework 2 Partial Solutions

The written portion of this week's homework will give you some practice working with the binary representation of integers and reasoning with invariants. You are strongly advised to review the C0 language reference guide (available at http://c0.typesafety.net/) for details on integer manipulation.

1. **Basics of C0**

(3) (a) Let x be an `int` in the C0 language. Express the following operations in C0 using only constants and the bitwise operators (`&`, `|`, `^`, `~`, `<<`, `>>`). Your answers should account for the fact that C0 uses 32-bit integers.

i. Set a equal to x, where the alpha and green components have both been set to 0, with the red and blue components left unchanged. (eg `0xAB12CE34` becomes `0x00120034`; see Section 1.1 of the Programming portion for more info)

Solution: `a = x & 0x00FFFF00;`

ii. Set b equal to x with its middle 16 bits flipped ($0 \implies 1$ and $1 \implies 0$) (eg `0xAB0F1812` becomes `0xABF0E712`)

Solution: `b = (x & 0xFF0000FF) | (~x & 0x00FFFF00);`

iii. Set c equal to x with its highest 8 bits set to 1 and with its lowest 8 bits set to 0. (eg `0xAB12CE34` becomes `0xFF12CE00`)

Solution: `c = (x | 0xFF000000) & 0xFFFFFF00;`

iv. Set d equal to x with its highest and lowest 16 bits swapped (eg `0x1234ABCD` becomes `0xABCD1234`)

Solution: `d = (x << 16) | ((x >> 16) & 0x0000FFFF);`

(1) (b) Are the following two `bool` expressions equivalent in C0, assuming `x` and `y` are of type `int`? Explain your answer.

`(x%y < 122) && (y != 0)`   `(y != 0) && (x%y < 122)`

Solution: They are not equivalent, as we can demonstrate with Coin:

```
--> int x = 12;
x is 12 (int)
--> int y = 0;
y is 0 (int)
--> (x%y < 122) && (y != 0);
Error: division by zero.
Last position: <stdio>:1.2-1.5
--> (y != 0) && (x%y < 122);
false (bool)
```

(1) (c) Is the following code a valid way to check if $a + b + c$ overflows? If not, give values for a, b and c such that the check will return an incorrect result:

```
bool safe_add(int a, int b, int c)
{
  if (a > 0 && b > 0 && c > 0 && a + b + c < 0) return false;
  if (a < 0 && b < 0 && c < 0 && a + b + c > 0) return false;
  return true;
}
```

Solution: This is not a valid way to check if $a + b + c$ overflows: if `a` = `b` = `c` = `2000000000`, the `safe_add(a,b,c)` will return `true` but adding the three numbers will result in `1705032704`.

(3) (d) For each of the following statements, determine whether the statement is true or false in C0. If it is true, explain why. If it is false, give a counterexample to illustrate why.

i. For every `int` x and y, $x < y$ is equivalent to $x - y < 0$

Solution: False, `x = int_min()`, `y = -1`.

ii. For every `int` x: x `>>` 1 is equivalent to $x/2$.

Solution: False, `x = -5`.

iii. For every `int` x, y, and z: $(x + y) * z$ is equivalent to $z * y + x * z$.

Solution: True, this is the distributivity law (page 4 of the lecture 3 notes).