15-122 : Principles of Imperative Computation

Summer 1 2012

Assignment 1

(Theory Part)

Due: Tuesday, May 29, 2012 in class

Name: ____________________________________________________________

Andrew ID: _______________________________________________________

Recitation: ________________________________________________________

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 should type up your solutions or write them neatly by hand, and you should submit your work in class on the due date just before lecture or recitation begins.

Be sure to staple your homework before you submit it and make sure your name and section letter is clearly shown.

<table>
<thead>
<tr>
<th>Question</th>
<th>Points</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>Total:</td>
<td>25</td>
<td></td>
</tr>
</tbody>
</table>
1. Basics of C₀

(4) (a) Let \( x \) be an \texttt{int} in the C₀ language. Express the following operations in C₀ using only constants and the bitwise operators (\&, |, \^, \~\, <\ll, \gg\>). [1 pt each]

i. Set \( a \) equal to \( x \) multiplied by 64.

Solution:

ii. Set \( b \) equal to \( x \mod 16 \), assuming that \( x \) is positive.

Solution:

iii. Set \( c \) equal to \( x \) with its middle 12 bits all set to 1.

Solution:

iv. Set \( d \) equal to the intensity of the green component of \( x \), assuming \( x \) stores the packed representation of an RGB color. The intensity should be a value between 0 and 255, inclusive.

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

\[(x/y < 122) \&\& (y \neq 0) \quad (y \neq 0) \&\& (x/y < 122)\]

Solution:

(c) 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 \(x\): \(x + 1 > x\).

Solution:

ii. For every int \(x\): \(x \ll 1\) is equivalent to \(x \ast 2\).

Solution:

iii. For every int \(x\): \(x/10 \ast 10 + x\%10\) is equivalent to \(x\).

Solution:

iv. For every bool \(a\) and bool \(b\): \(a \&\& !b\) is equivalent to \(!(!a \mid \mid b)\).

Solution:
2. **Reasoning with Invariants**

The Fibonacci sequence is shown below:

1, 1, 2, 3, 5, 8, 13, 21, 34, 55, ...

Each integer in the sequence is the sum of the previous two integers in the sequence. Consider the following implementation for `fastfib` that returns the \(n\)th Fibonacci number (the body of the loop is not shown).

```c
int fib(int n)
//@requires n >= 1;
{
    if (n <= 2) return 1;
    else return fib(n-1) + fib(n-2);
}
int fastfib(int n)
//@requires n >= 1;
//@ensures \result == fib(n);
{
    if (n <= 2) return 1;
    int i = 1; int j = 1;
    int k = 2; int x = 3;
    while (x < n)
        //@loop_invariant 3 <= x && x<=n && i==fib(x-1) && j==fib(x-2) && k==i+j;
    {
        // LOOP BODY NOT SHOWN
    }
    return k;
}
```

(a) Using the precondition and loop invariant, reason that the `fastfib` function must return the correct answer, even if you don’t know what is in the body of the loop. You may assume the loop invariant is correct.

---

**Solution:**
(2) (b) Based on the given loop invariant, write the body of the loop.

Solution:

(2) (c) What is the largest Fibonacci number that can be generated by your fastfib program before overflow occurs? (Overflow occurs when you add two positive integers together and get a negative result, or when you add two negative integers together and get a positive result.) Show how you derived your answer.

Solution:
3. More on Reasoning with Invariants

A C₀ programmer was writing a function to add up the first n odd positive integers and, after testing, noticed that the first n odd positive integers always seemed to add up to $n(n - 1)/2$. To verify this, the programmer added annotations to the function as shown below:

```c
int sum_first(int n)
//@requires n > 0;
//@ensures 2 * \result == n * (n - 1);
{
    int sum = 0;
    int i = 0;
    while (i < n)
        //@loop_invariant 0 <= i && i <= n;
        //@loop_invariant 2 * sum == i * (i - 1);
        {
            sum = sum + i;
            i = i + 1;
        }
    return sum;
}
```

Prove that the postcondition (ensures) holds for the function using the given precondition (requires) and the loop invariants:

1. (a) Give a brief argument explaining why the loop must terminate.

Solution:

1. (b) Show that each loop invariant is true immediately before the loop condition is tested for the first time.

Solution:
(c) Show that if each loop invariant is true at the start of a loop iteration, then the loop invariants are also all true at the end of that iteration.

Solution:

(d) Show that if the loop terminates, the postcondition must hold.

Solution: