15-122: Principles of Imperative Computation — Spring 2023

Lab 02: What's the Point — Tuesday January 17th

Collaboration: In lab, we encourage collaboration and discussion as you work through the problems. These activities, like recitation, are meant to get you to review what we've learned, look at problems from a different perspective and allow you to ask questions about topics you don't understand. We encourage discussing problems with other students in this lab!

The Integer Logarithm

The *(base 2) integer logarithm* of a strictly positive number x is the largest integer r such that $2^r \leq x$. The function `ilog` below computes the integer logarithm of its input.

```
1  int POW2(int x)      // computes 2^x
2  //@requires x >= 0;
3  {
4    if (x == 0) return 1;
5    return 2 * POW2(x-1);
6  }
7
8  int ilog(int x)      // returns the largest r such that 2^r <= x
9  //@requires x > 0;
10 //@ensures \result >= 0;
11 //@ensures POW2(\result) <= x;
12 {
13   int a = x;
14   int r = 0;
15   while (a > 1)
16   //@loop_invariant a >= 1;
17   //@loop_invariant r >= 0;
18   //@loop_invariant a * POW2(r) <= x;
19   {
20     a = a/2;
21     r++;
22   }
23   //@assert a == 1;
24   return r;
25 }
```

Notice the specification function `POW2(x)` which computes 2^x. You may assume it is correct without referring to its code.

In this lab, we will show that `ilog` is correct, i.e., that its postconditions are true whenever its preconditions hold.[1] Recall that, for a function with a single loop, we go through the following steps to prove correctness:

- The loop invariants are valid, i.e., each is true INITially and is PREServed by an arbitrary iteration of the loop.

[1] Observe however that the postconditions of `ilog` do not fully match what we expect from a function that computes the integer logarithm of its input x. Specifically, it does not ensure that the returned value is the *largest* r such that $2^r \leq x$. But there is only so much we can do in one lab ...

- The loop TERMinates.
- The loop invariant and the negation of the loop guard entail the postcondition when we EXIT the function.

INIT, PRES and EXIT are proved by point-to reasoning. We will focus on those!

INIT

Let's prove INIT for one of the loop invariants (the others are similar and you are encouraged to prove them on your own).

(2.a) Using point-to reasoning, prove that the first loop invariant, on line 16, holds initially.

To Show: `a >= 1` before any iteration of the loop

A. _____ by _____

B. _____ by _____

C. _____ by _____

D. `a >= 1` by math on (C)

1.5pt

EXIT

Let's continue with EXIT (in what order your prove these parts doesn't matter since you need to prove them all anyway). Our function has two postconditions, so we'll have two proofs.

(3.a) Using point-to reasoning, complete the proof that the first postcondition, on line 10, holds.

To Show: `\result >= 0`

A. `\result == r` by line 24

B. _____ by _____

C. `\result >= 0` by _____

(3.b) Proving that the second postcondition, on line 11, is more involved as it requires doing some math. But you can do it! *(You may not need all lines provided.)*

To Show: `POW2(\result) <= x`

A. `\result == r` by line 24

B. _____ by _____

C. _____ by _____

D. _____ by _____

E. _____ by _____

F. _____ by _____

G. `POW2(\result) <= x` by _____

PRES

Preservation at last!

(4.a) Complete the proof that the second loop invariant, on line 17, is preserved by an arbitrary iteration of the loop.

Assuming: `r >= 0`,
To Show: `r' >= 0`.

A. `r >= 0` assumption

B. _____ by _____

C. _____ by _____

D. `r' >= 0` by _____

3pt

(4.b) The biggie is the third loop invariant on line 18. Complete the following proof that it is preserved by an arbitrary iteration of the loop. *(You may not need all lines provided.)*

Assuming: _____,
To Show: _____.

A. _____ assumption

B. `(a/2) * 2 <= a` by math (since `a` is positive — line 16)

C. _____ by _____

D. _____ by _____

E. _____ by _____

F. _____ by _____

G. _____ by _____

H. _____ by _____

I. _____ by _____

J. _____ by _____

K. _____ by _____

4pt

For additional practice, you may want to prove on your own that the first loop invariant, on line 16, is also preserved.

TERM

We also need to prove that the loop terminates. Feel free to do so at your leisure.