COMP 211: Lab 4
Fall, 2014

Task: Fill out the survey `https://www.surveymonkey.com/s/comp211`.

1 Proving Loop Invariants

Recall the code from yesterday's lecture:

```
int POW(int x, int y)
//@requires y >= 1;
{
  int r = 1;
  for (int i = 0; i < y; i = i + 1) {
    r = x * r;
  }
  return r;
}

int f (int x, int y)
//@requires y >= 1;
//@ensures \result == POW(x,y);
{
  int b = x;
  int e = y;

  int r = 1;
  while (e > 1)
  //@loop_invariant r * POW(b,e) == POW(x,y);
  {
    if (e % 2 == 1) {
      r = b * r;
    }
    b = b * b;
    e = e / 2;
  }

  return r * b;
}
```

Task Fill in this table for `f(2,6)`, where row i contains the values of the variables after i times through the loop body.

	x	y	x^y	b	e	r	$r*b^e$
0							
1							
2							

In general, to prove that a function (with one loop in it) meets its contract, we do the following:

1. Assume that the pre-condition (ensures) holds.
2. Prove that the loop invariant is always true. This consists of two steps.
   - **Init**: Show that the loop invariant holds *initially* just before the loop is run (specifically, just before the loop exit test is run for the first time).
   - **Preservation:** Assume the loop invariant is true at the beginning of an *arbitrary* time through the loop body. Prove that the loop invariant is true at the end of one execution of the loop body. You can also assume that the loop exit test evaluates to true while showing preservation (because otherwise the loop body wouldn't be running).
3. Show that the loop invariant implies the post-condition. You can also assume that the loop exit test evaluates to false after the loop (because otherwise you would still be executing the body of the loop).

In the first part of this lab, you will work through the proof from yesterday's lecture to make sure you understand it.

First, we do Step 2, and show that the loop invariant is always true.

- **Init** We did this in class: before the loop `b` is `x` and `e` is `y` and `r` is 1, so `r * POW(b,e) == 1 * POW(x,y) == POW(x,y)`.
- **TASK: Preservation** On the next page, finish the proof that the loop invariant is preserved. You may use the following two lemmas:

```
Lemma 1: If e is even, then POW(b,e) == POW(b*b, e / 2)

Lemma 2: If e is odd, then POW(b,e) == b * POW(b*b, e / 2)
Proof:
When e is odd, e == 2k+1, where
k == e / 2 (because integer division rounds down).  Therefore

     POW(b,e)
  == POW(b,2k+1)          by definition of e
  == b * POW(b,2k)        by definition of POW
  == b * POW(b * b, k)    by Lemma 1
  == b * POW(b * b, e/2)  by definition of k
```

```
Assume the loop invariant holds at the beginning of the loop:
  r * POW(b,e) == POW(x,y)                              (LI)
We need to show that
  r' * POW(b',e') == POW(x,y)                           (LI')
where r' b' e' are the values of r, b, and e at the end of
one execution of the loop body.
The proof requires different cases for when e is even and when e is odd.

Case where e is even:
  In this case,

    r' == ______________________________

    b' == ______________________________

    e' == ______________________________

  Therefore
       r' * POW(b',e')

    == ____________________     by definition of r' b' e'

    == r * POW(b,e)             by _________________________________

    == POW(x,y)                 by _________________________________


Case where e is odd:
  In this case,
    r' == ______________________________

    b' == ______________________________

    e' == ______________________________

  Therefore
       r' * POW(b',e')

    == ___________________________  by definition of r' b' e'

    == r * POW(b,e)                by _________________________________

    == POW(x,y)                    by _________________________________
```

Have someone check your work up to this point!

Task Next, we will work on step 4: showing that the loop invariant implies the post-condition (ensures).

The proof we gave in class went as follows: We need to show that

```
\result == POW(x,y)
```

Since `f` returns `r*b`, it suffices to show that

```
r * b == POW(x,y)
```

after the loop.

Because we have proved that the loop invariant holds after the loop, we know that after the loop

```
r * POW(b,e) == POW(x,y)
```

Thus, if we can show that `e` is 1 at the end of the function, then we can prove the result as follows:

```
   r * b
== r * POW(b,1)    because POW(b,1) == b
== r * POW(b,e)    because e == 1
== POW(x,y)        by the loop invariant.
```

How do we prove that `e == 1` after the loop? The intuition is this: We know that `e >= 1` at the start of the function. If `e == 1` at the start, then we don't run the loop, and `e == 1` at the end. If `e > 1`, then we do run the loop, and running loop divides `e` by 2, so it will eventially reach 1.

However, this intution involes saying words like "running the loop divides `e`" and "eventually", which is sure-fire sign that we *need another loop invariant!*.

The key idea is that we can "pinch" `e` as follows. When the loop exits, we know that the loop exit test is false, i.e. `!(e > 1)`, because otherwise we would still be running the loop. If we also knew that `e >= 1`, then we would know that `e == 1`, by process of elimination. So we can use an additional loop invariant that `e >= 1`. The fact that the loop test is true in the body of the loop will be important to proving that this loop invariant is preserved.

Loop invariant: e >= 1;

Init: The loop invariant holds just before the loop is
executed because

_____

Preservation:
Assume _____ holds before the loop is run.

To show: _____

By definition, e' == _____

Because the loop test is true, _____

Therefore _____

Because we have shown init and preservation, the loop invariant holds at
the end of the loop, so we know that, after the loop,

_____

Moreover, after the loop, the loop exit test must evaluate to
false, so

_____

Therefore e == 1 at the end of the function.

Have someone check your work up to this point!