CPS122 - OBJECT-ORIENTED SOFTWARE DEVELOPMENT

An Example of a Correctness Proof for a Loop

The following code raises a specified double to a specified integer power - e.g. power(1.5, 3) would return $1.5^3 = 3.375$.

```
// Precondition: exponent ≥ 0, base != 0.0
// Postcondition: returned value is base^exponent

public static double power(double base, int exponent)
{
     double v = 1.0, b = base;
     int e = exponent;

     while (e > 0)
     {
          if (e % 2 == 1)
          {
               v *= b;
               b *= b;
               e /= 2;
          }
          else
          {
               b *= b;
               e /= 2;
          }
     }

     return v;
}
```

Correctness Proof

To prove the correctness of the above, we make use of the following loop invariant:

```
e ≥ 0 and b != 0 and v * b^e = base^exponent
```

1. The following annotated code demonstrates that the invariant is established and preserved:

```
-- prologue comments and function prototype omitted

// exponent ≥= 0 and base != 0 (precondition)

double v = 1.0, b = base;
int e = exponent;

// e ≥ 0 and b != 0 and v * b^e = base^exponent (invariant is established)

while (e > 0) // Invariant: e ≥ 0 and b != 0 and v * b^e = base^exponent
{
    // e > 0 (from loop condition) and b != 0 and v * b^e = base^exponent
```

```
    if (e % 2 == 1)
    {
         // e is odd and e > 0 and b != 0 and v * b^e = base^exponent

         v *= b;

         // e is odd and e > 0 and b != 0 and v * b^(e-1) = base^exponent

         b *= b;

         // e is odd and e > 0 and b != 0 and v * b^((e-1)/2) = base^exponent

         e /= 2;   // Since e is an odd integer, the result of dividing e by 2
                   // using integer division is (e-1)/2

         // e ≥ 0 and b != 0 and v * b^e = base^exponent
    }
    else
    {
         // e is even and e > 0 and b != 0 and v * b^e = base^exponent

         b *= b;

         // e is even and e > 0 and b != 0 and v * b^(e/2) = base^exponent

         e /= 2;   // Since e is an even integer, the result of dividing e by 2
                   // using integer division is (e/2)

         // e ≥ 0 and b != 0 and v * b^e = base^exponent
         // (invariant is preserved)
    }
}

// e = 0 and b != 0 and v * b^e = base^exponent

-- see below
```

2. To prove <u>loop termination</u>, note that e is an integer that is ≥ 0. Each time through the loop, we divide e by 2. This must eventually result in e becoming zero, at which point the loop terminates.

3. We can now show that the <u>postcondition</u> for the function is established.

```
-- loop body and terminating condition as above

// Since b^0 = 1 for any non-zero value of b, the loop terminating
// condition is equivalent to v * 1 = base^exponent, or v = base^exponent


return v;

// returned value is base^exponent
```