Principles of Software: Homework #2

Due on March 5 2021

Prof. Valera

Jared Gridley

Problem 1

Problem 1 (15 pts): Product by addition

a) Find a suitable loop invariant. (3 pts)

```
invariant x * y + result = n * m
```

b) Show that the invariant holds before the loop (base case). (1 pts)

```
Before going into the loop, we know that y = n and n >=0 (as given by the precondition
and assignment). We know that the base case is when result = 0, so then the invariant
condition will be x * y + 0 = n * m, so x*y=m*n, which will hold because we set x = m
and y =n.
```

c) Show by induction that if the invariant holds after k-th iteration, and execution takes a k+1-st iteration, the invariant still holds (inductive step). (6 pts)

```
By Induction show that k -> k+1
    If we assume that all the condition for Prod(k) held before the loop, then x,y can be
    altered in 2 ways.
        If y is even, x = x+x and y = y/2
            So we know that x*y+result = n*m by our assumption, so substituting in the new
            values, we get (2x)(y/2) + result = n*m --> 2xy/2 + result = n*m -->
            x * y + result = n * m, so the invariant holds throughout this alteration.
        If y is odd, result = result + x and y = y-1
            So plugging in the new value, x(y-1) + result + x = n*m -->
            xy - x + result + x = n*m --> xy+result + (x-x) = n*m --> xy+result = n*m,
            so the invariant will also hold through this alteration.
    Since each loop alteration will maintain the invariant condition for k -> k+1, then
    it must be true k+1-st iteration.
```

d) Show that the loop exit condition and the loop invariant imply the postcondition result = m*n. (1 pts)

```
We know that exit condition is y=0 as a result of the loop invariant so combining them we get that
```

e) Find a suitable decrementing function. Show that the function decreases at each iteration and that when it reaches a minimum the loop is exited. (2 pts)

```
D = y (in the code it becomes decreases y)
```

Problem 2

Problem 2 (14 pts) The Simplified Dutch National Flag Problem

a)

```
int dutch(Array arr){
    var i = 0;
    var j = 0;
    while (i < arr.Length){
        if (arr[i] == 'r'){
            if ( i != j) {
                swap(arr, i, j);
                }
            j = j + 1;
        }
        i = i + 1;
        }
    k = j+1;
    return k;
    }
```

b) Write an expression for the postcondition. (2 pts)

```
Postcondition: 0 <= k <= arr.Length
```

c) Write a suitable loop invariant for all loops in your pseudocode. (4 pts)

```
Invariant: j <= i
```

Problem 3

Problem 3 (20 pts): Additive Factorial

```
function Factorial(n: int): int
    requires n >= 0
    {
    if n == 0 then 1 else n * Factorial(n-1)
    }

method LoopyFactorial(n: int) returns (u: int)
    requires n >= 0
    ensures u == Factorial(n)
    {
    u := 1;
    var r := 0;
    while (r < n)
      invariant Factorial(r) == u
    {
      var v := u;
      var s := 1;
      while (s<=r)
        invariant s*v == u
      {
        u:=u+v;
        s:=s+1;
      }
      r:=r+1;
      assert Factorial(r) == u;
    }
  }
```

b) 2 points – Proof for the base case of inner loop

```
Before the loop we know that they values in the invariant condition are s=1,
and v=u=1, so we have 1*1 = 1, so this hold as 1=1.
```

c) 3 points – Proof for the inner loop induction

```
Prove by induction for kth iteration --> k+1th iteration:
    Assume that the conditions for the kth iteration are correct, that is
    s*v == u.
    So for k=1, we know that the alterations will be that u=u+v and s=s+1, so
    we can modify the invariant condition for k+1 --> (s+1)v = u + v
    --> sv + v = u + v. We can eliminate the parts from our assumption because
    we assume them to be true, and we are left with v = v which is an identity
    so its true, and the invariant holds for k+1th iteration
Thus, if we have proved through induction that the inner loop invariant holds.
```

d) 2 points – Proof for the outer loop base case

```
The base case values are n = 0 as it is the beginning of the range of possible
values for n, and u = 1. We van then verify this through the condition u==Factorial(n)
--> 1 == Factorial(0) --> we know that the factorial of 0 is 1  so we are left with 1==1
which is true.
```

e) 3 points – Proof for the outer loop induction

```
Prove by induction for kth iteration --> k+1th iteration: (Outer Loop)
    To prove this we can assume that the outer loop invariant condition holds for the
    kth element, while using the inner loop invariant we proved in c.
        For k + 1, we can use our proof from part c, to show that kth_itr --> k+1th_itr and
        that s*v ==u, holds true.
        For the outer loop invariant (exit condition) Factorial(r+1) --> (r+1)*Factorial(r)
        which we can combined with our inner loop to get the value for u+1:
        (r+1)Factorial(r) == r(u+v), (u+v is is multiplied by r because it goes through r
        iterations of v being added to u in the inner loop). From the outer loop invariant
        condition we can simplify it to (r+1)u==r(u+v) --> ru + u = ru + v --> u == v,
        which we know to be true because we set v equal to u in the first line.
    Thus, by induction, the invariant condition Factorial(r) == u holds true for the
    k+1th iteration.
```