CSC238:Discrete Mathematics for Computer Science – Spring 2003

Assignment 2: Program Correctness.

Due: 10 a.m., Thurs., Feb. 13

This assignment is worth 10 percent of the total marks for this course.

Here we consider proving programs are correct and estimating their time complexity. We will mark only a (secret) subset of the questions below. As always in this course, your proofs will be marked for correctness, along with brevity and clarity.

Your answers can be handwritten. Use standard 8.5 by 11 inch paper. Please staple all the sheets together, and hand them to your tutor at the **beginning** of the tutorial on the due date. If you cannot make it to that tutorial, then leave your assignment at your instructor's office (Pratt, Room 283) before 10am on the due date. Since the tutors will be discussing the solutions in the tutorial immediately after your assigments are due, we will not accept late assignments (the course homepage describes what you should do in case of medical or other emergencies which prevent you from completing an assignment on time).

1. Prove that the `Mod(`x`,` m`)` program defined below is correct.

   `Mod(`x`,` m`)`
   Precondition: x, m are natural numbers, $m > 0$.
   Postcondition: `Mod(`x`,` m`)` returns a natural number r,
   with $0 \leq r < m$, such that there exists an integer n with $x = nm + r$.

   ```
   r := x
   while r ≥ m do
       r := r − m
   end while
   return r
   ```

2. Use a loop invariant to prove that the following program is correct.

   Precondition: x is natural number.
   Postcondition: $y = 0$.

   ```
   y := x ∗ x
   while y ≠ 0 do
       x := x − 1
       y := y − 2 ∗ x − 1
   end while
   ```

3. Prove that the loop below terminates if the precondition is satisfied before the loop starts.

Precondition: x, y are natural numbers, and x is even.

```
while x ≠ 0 do
    if y ≥ 1 then
        y := y − 3
        x := x + 2
    else
        x := x − 2
    end if
end while
```

4. Consider the following precondition, postcondition pair:

   `BeLow(`A`, `n`, `y`)`
   Precondition: $n \geq 1$ and A is an array of length n sorted in non-decreasing order. That is, for any integer i with $1 \leq i < n$, we have $A[i] \leq A[i+1]$. Finally, y is an integer such that $y \leq A[n]$.
   Postcondition: `BeLow(`A`, `n`, `y`)` returns the minimum integer $k \geq 1$ such that $A[k] \geq y$.

   (a) Write a binary search style algorithm for solving this problem. That is, your algorithm **must** run in $O(\log(n))$ time, like binary search does. Algorithms which require time $\Omega(n)$ will receive a mark of 0. (Hint: First write a loop invariant for your program, and then write the loop body.)

   (b) Write a loop invariant for your program and use it to prove that your program is correct.

   (c) Prove that your program executes in $O(k)$ steps when the length of A is $n = 2^k$.

5. Show that for every real number $d > 0$, the function $f_d(n) = \sum_{k=0}^{n} k^d$ satisfies $f_d(n) \in \Theta(n^{d+1})$.

6. Let k be an integer and let $f(n)$ and $g(n)$ be positive, real-valued functions defined for natural numbers $n \geq k$ (that is, $f(n), g(n) > 0$ for all $n \geq k$). Suppose $\lim_{n\to\infty} f(n)/g(n) = x$ for some real number $x < \infty$.

   (a) Prove that $f(n) \in O(g(n))$.

   (b) Prove that if $x \neq 0$ then $f(n) \in \Theta(g(n))$.

   (c) Prove that if $x = 0$ then $g(n) \notin O(f(n))$.

   (d) Prove $\log(\log(n)) \in O(\log(n))$ but $\log(\log(n)) \notin \Theta(\log(n))$ using parts (a-c) above.

   (e) Does the limit $\lim_{n\to\infty} f(n)/g(n)$ exist whenever $f(n) \in \Theta(g(n))$? Explain.