CSC236 – Problem Set 7

There are two components of this problem set: a preliminary question designed to check your understanding of the basic topics covered this week, and a set of more challenging questions designed to make you think critically about the material and apply it in new contexts. Get in the habit of starting work early – the less time you give yourself, the most stressed you’ll find yourself each week!

Caution: you must submit two separate files in two separate locations on MarkUs, one for the Preliminary and one for the Challenge.

To avoid suspicions of plagiarism, clearly state any resources (people, print, electronic) outside of your group, the course notes, and the course staff, you consulted at the beginning of your assignment submission.

Preliminary: due March 4, 2014 8:00 pm

This question is an opportunity for you to check your understanding of the topics and practice writing formal solutions. This is a valuable learning opportunity – if you see that you’re at a loss, get help quickly!

Your goal should not be to get the right answer, but to convince the marker that you know what you’re doing. This question is marked on the following 3-point scale:

<table>
<thead>
<tr>
<th>3: You’ve mastered this topic</th>
<th>1: You don’t really know what you’re doing</th>
</tr>
</thead>
<tbody>
<tr>
<td>2: You’re almost there, but missing something</td>
<td>0: You didn’t submit/had absolutely no clue</td>
</tr>
</tbody>
</table>

This question must be completed INDIVIDUALLY.

Consider the following code.

```python
1 def mystery(a, b):
2     ''' Pre: a and b are positive integers. '''
3     x = a
4     d = 0
5     while x >= b:
6         x -= b
7         d += 1
8     return d
```

Prove the following loop invariant for the while loop: $x \geq 0$ and $a = x + b \times d$. Make sure you’re specific in describing how the code changes the variables.

Note that to prove the invariant, you need “steps 1 and 2” from lecture: prove that the first time the loop is reached, the invariant holds, and then prove that if the invariant holds at the beginning of an iteration, it still holds at the end of the iteration.
Challenges: due March 8, 2014 noon

Answer each question completely, always justifying your claims and reasoning. Your solution will not just be graded on correctness, but on its clarity as well. Technically correct answers that are hard to understand will not receive full marks. Mark values for each question are contained in the [square brackets].

You may work in groups of up to THREE to complete these questions.

1. Recall the mystery function from the Preliminary.
   (a) [1] State a good postcondition for mystery (your postcondition must be specific enough to describe what the function does, and not just say something like “d > 0”.
   (b) [2] Using the loop invariant given in the Preliminary, prove that mystery is correct according to the pre- and postconditions. You do not need to prove termination.

2. [6] Consider the following code for insertion sort:

   ```python
   def insertion_sort(A):
   Pre: A is a list of numbers
   Post: A is sorted in non-decreasing order
   i = 1
   while i < len(A):
     j = i
     while j > 0 and A[j] < A[j-1]:
       swap A[j-1], A[j]
       j -= 1
     i += 1
   ```

   The loop invariant for the outer loop is quite simple: 0 < i ≤ len(A) and A[0..i-1] is sorted in non-decreasing order. Clearly, once the loop finishes and i = len(A), this will imply the postcondition, that A is entirely sorted.

   State and prove the loop invariant for the inner loop that can be used to prove the loop invariant for the outer loop. (As in the preliminary, this amounts to steps 1 and 2 from lecture). However, you do not need to prove how it is connected to the outer loop (though this is good practice). Don’t forget to bound the value of j in your invariant!

   Hint: You should start by fixing a value for i, and assuming the outer loop invariant holds for that i. You should use this assumption in your proof for the inner loop invariant. The goal of the inner while loop is to “insert A[j]” into the right place in A[0..i].

3. [6] Consider the following problem: given a binary string s, determine the length of the longest prefix of s that has more 0’s than 1’s. For example for the string “0010111010”, the longest prefix of s has that more 0’s than 1’s is “00101,” which has length 5.

   Write an iterative algorithm that solves this problem. Its signature and specifications are given below. You may treat s like a list, and use the same syntax. Your algorithm may not be recursive, and should use at least one loop. You aren’t required to prove that your code is correct, but be sure to include comments to help the markers understand your code.

   This exercise is a chance to design an iterative algorithm using invariants: in addition to the code, state (but don’t prove) a suitable loop invariant for each loop you write!

   ```python
   def pre_zeros(s):
   Pre: s is a binary string (possibly empty)
   Post: returns the length of the longest prefix of s that has more 0’s than 1’s.
   ```

   ```python
   <YOUR CODE GOES HERE>
   ```