Assignment 4 (Written, Tool): Hoare Logic and Spec#

17-654/17-754: Analysis of Software Artifacts
Jonathan Aldrich (jonathan.aldrich@cs.cmu.edu)

Due: Thursday, February 22, 2007 (10:30am)

150 points total

Turn in a file named <username>-17654-A4.zip, where username is your Andrew id. This zip file should contain answers.{txt, pdf, doc} which should contain your responses to each of the written questions. The zip file should also contain Stack.ssc, stack-output.txt, Min.ssc and min-output.txt which are described in the following sections. At the top of the answers file, state your name, Andrew id, and how long you spent on the assignment.

Assignment Objectives:

- Become familiar with specification constructs including pre- and post-conditions, loop invariants and variant functions
- Prove small programs correct using Hoare logic techniques
- Use Spec# to check functional correctness properties of programs

1 Hoare Logic (50 points)

Question 1 (10 points).

Consider the following WHILE program:

$i := 1;$
$r := 1;$
$while\ (i < n)\ do$
  $i := i + 1;$
  $r := r + i$

For the while program given above, state a (a) precondition, (b) postcondition, (c) loop invariant, and (d) variant function. The postcondition should precisely define the value of r in terms of n.

Question 2 (30 points).

Using weakest preconditions, state the proof obligations that assure (a) the invariant is initially true on entry to the loop, (b) the loop invariant holds after execution of each loop, (c) the variant function is greater than zero at loop entry, (d) the variant function decreases in the loop, and (e) the postcondition holds.

Question 3 (10 points).

Prove each of the proof obligations above. Your proof should be done at the level of detail shown in lecture, i.e. small steps with justifications given for each step.

2 Using Spec# (100 points)

Question 4 (60 points).

For this assignment you will be required to use the Microsoft Visual Studio environment. Visual Studio 2005 works. Visual Studio 2003 should work, but you should consider it unsupported for the purposes of this course. If you do not currently have Visual Studio, you are all able to obtain it through ISR. According to Ed Walter, you have all received instruction on how to download and install the software that is provided as part of the Microsoft Academic Alliance. If you have trouble with this, please see him for assistance. If you do not use a Windows-compatible PC, it will be necessary to obtain access to one for this assignment. The clusters are a good place for this. All instructions from here on assume that you have Visual Studio installed.

Installing Spec# and the Simplify Theorem Prover: In order to install Spec#, start by visiting the Spec# homepage:

`http://research.microsoft.com/specsharp/`

From here you will be able to install Spec# for VS2005. Do **not** use the VS2003 version of Spec#. Additionally, you must install the Simplify theorem prover. Instructions on this process are given at the following URL:

`http://research.microsoft.com/specsharp/simplify.htm`

From within Visual Studio, open the project Stack that we have included with the assignment. This project contains C# source files, Stack.ssc and StackCheck.ssc. The Spec# static verifier should be run automatically in the background. Any warnings that it finds will be printed to the "Error List" window. (If this window is not visible, enable it via the "View" menu.) The output of the static verifier can also be seen in the "Output" window, and you will need to have this window visible so that can copy the tool's output and submit it.

Add pre- and post-conditions and invariants to Stack.ssc and fix any bugs you find in the code, until Spec# runs on both files without producing any warnings. You may not edit StackCheck.ssc. Nor may you remove the annotations that are already present in Stack.ssc.

(To learn more about the specifications that you can use in Spec# we have included some documentation in a file called "specsharp_docs.html." The installation of Spec# includes sample files which may also be helpful. Finally, additional documentation can be found on the Spec# web site.)

Turn in: (a) your edited version of Stack.ssc, and (b) a printout of Spec#'s output when run on the two files in stack-output.txt.

Question 5 (30 points).

Write a method that returns the minimum integer in a given array. This method should have the following signature:

```
public int findMin(int[]! array);
```

Check the correctness of this method using the Spec# static verifier, i.e. that it really finds the minimum value. Your solution should use appropriate requires, ensures, and invariant clauses.

Turn in: (a) your C# code Min.ssc, and (b) a printout of Spec#'s output when run on the Min.ssc in min-output.txt.

Question 6 (10 points).

In the file answers.txt, criticize the Spec# static verifier tool (1-2 paragraphs). What did you like about it, and conversely what stands in the way of making this a practical tool?