CpSc 3720 Project 4

Due: Tuesday, April 19

Objective
The objective is to enhance your understanding of formal methods in software engineering and experiment with a system that implements them.

Collaboration Policy
You may work independently or with one partner. If you work with a partner, you can do problem 1 together, then one partner should independently work problem 2, and the other should independently work problem 3. You may assist each other with debugging and checking the proofs after working independently. You will do a joint submission of all three problems, and will receive a single grade.

You may not get help from anyone not on your team.

Problem 1: Top (10 points)
1. Create a login for the Resolve IDE by navigating to the following page and clicking Register:

   http://resolve.cs.clemson.edu/teaching/

2. Login using your credentials and create an empty Facility. To create a facility, click the Components button. Then, right-click on Programs and choose New Facility. Enter the name username_Project4 for your facility (replace username by your Clemson username). Copy and paste the contents of the Project 4 facility provided on the course website Project page into your new facility.

3. Review the specifications for Stack_Template and Queue_Template by clicking the Components button, then navigating to Concepts > Stack_Template > Stack_Template. This will open a new tab with the spec for Stack_Template. Repeat the same procedure, this time for Queue_Template. At this point, you should have three tabs open: One for your facility, and the others with the Stack and Queue templates for your reference.

4. In your Project4 facility, find the Top operation. Review its specification carefully to ensure that you understand what it is supposed to do. Then, review the implementation. It contains a logic error. It also lacks sufficient preconditions. Fix both of these problems.
5. Comment out everything except the Top operation and the Facility statement just above it that instantiates the Stack_Template component. Now, click the **MP-Prove** button. If you have correctly fixed all of the bugs, there should be a green check mark beside all of the VC’s.

**Problem 2: Read_upto (20 points)**

This method should replace the contents of a Queue with values read from the keyboard (using the `Read()` function). The Count parameter determines how many values are to be read into the queue, and can be any number, bounded by the queue capacity.

The operation contains a logic bug and insufficient preconditions.

1. Fix the logic bug and add appropriate preconditions. Do not change postconditions or parameter modes.
2. Write a suitable loop invariant and fix the decreasing clause.
3. Click **MP-Prove**. For any VCs that the compiler cannot verify, copy the complete VC (showing the VC number, description, line number, goal, and givens), then write a short proof showing that the VC holds. Give a single proof for duplicate VCs. See below for requirements on your proof document.

**Problem 3: Remove_last (20 points)**

This method should remove the item on the bottom of a (nonempty) stack.

The operation contains a logic bug and insufficient preconditions.

1. Fix the logic bug and add appropriate preconditions. Do not change postconditions or parameter modes.
2. For both loops, write suitable loop invariants and fix the decreasing clauses.
3. Click **MP-Prove**. For any VCs that the compiler cannot verify, copy the complete VC (showing the VC number, description, line number, goal, and givens), then write a short proof showing that the VC holds. Give a single proof for duplicate VCs. See below for requirements on your proof document.
**Bonus (+10%)**

Write a complete table-style proof of correctness for the corrected Read_upto procedure. Use the format of the following example (you do not need to use the special markup):

http://people.cs.clemson.edu/~sschaub/cpsc3720/notes/CorrectnessOfClear.pdf

You must do this independently, even if working with a partner.

**Proof Document**

For problems 1-3 above, VC’s that do not prove by the Resolve automatic prover must be proved manually in a separate proof document. The proof document should be organized by problem (show the proofs for problem 1 in a section labeled “Problem 1”, etc.).

For each failing VC, paste the line of source code for the failing VC, copy the complete VC (showing the VC number, description, line number, goal, and givens), then write a short proof showing that the VC holds. Your proof should follow the format shown in the Verifying Software Components book on p. 104 under Table 5. If your proof is not immediately obvious, it will be counted wrong.

Some VC’s are duplicates; write a single proof for those.

**Submission**

- Turn in a title page, printout of your code listing, and a document with your proofs, stapled in that order. Teams should submit a single, joint document. The title page should indicate which team member did problem 2, and which did problem 3. Staple the bonus solution, if done, to the back.
- One team member should use the Export button in the web IDE to export a .json file containing your code. Submit it using the handin.cs.clemson.edu website.
- Late submissions will not be accepted.