CS 6110, Spring 2022, Assignment 6

Given 3/4/22 – Due 3/15/22 by 11:59 pm via your Github

NAME: **UNID:**

CHANGES: Please look for lines beginning with underlined words when they are made. none yet.

Answering, Submission: Have these on your private Github: a folder Asg6/ containing your submission, which in detail comprises:

- A clear README.md describing your files.
- Files that you ran + documentation (can be integrated in one place).
- A high level summary of your cool findings + insights + learning – briefly reported in a nicely bulletted fashion in your PDF submission.

Start Early, Ask Often! Orientation videos and further help will be available (drop a note anytime on Piazza for help).

I encourage students constructing answers jointly! *But that does not mean copy solutions, but discuss the question plus surrounding issues.*

1. (40 points) I've checked in Logic[1-5].als. There are two parts to this assignment:
   (a) (20 points)
      i. (10 points) Go through what I've provided in the above models, and for each file, produce a gist of the encodings I've done and a two-line comment saying how the encoding works. (Save Logic4.als for the next part.)
      ii. (10 points) Get into Logic4.als and make sure that `f2` indeed is encoding a general two-ary function. To "make sure," you must demonstrate a sufficient number of alloy tests/checks/runs and/or model-examination. Have at least 3-4 convincing demos of checks you came up with.
   (b) (20 points)
      i. (10 points) Lecture 16, Slide 4, Image of question 13(b): Encode this assertion similar to the encoding in Logic5.als and check for validity.
      ii. (10 points) Lecture 16, Slide 4, Image of question 13(c): Encode this assertion similar to the encoding in Logic5.als and check for validity.

   (a) (20 points)

      i. (10 points)

      - **Logic 1**
      - **Logic 2**
      - **Logic 3**
      - **Logic 4**
      - **Logic 5**

2. (40 points)
   (a) (20 points)
      i. (10 points) Go through Mike-Gordon-Slides.pdf and also read Mike Gordon's book, making notes about the proof rules of Hoare Logic presented there. Include assignment, if, for, while, precondition strengthening, and postcondition weakening.

ii. (10 points) Get Dafny and Verifast installed. Please report issues you faced. Say which platform. Put notes in the tools GDoc to help each other.

(b) (20 points) Run the first two Dafny exercises in Lec16.pdf and ask questions on Piazza. All details are given to you.

(a) **20 points**

i. **(10 points) Mike Gordon's slides**

ii. **(10 points) Dafny verifast installation**

- **Verifast installation**
  - Verifast can be installed by downloading a tar file from `https://github.com/verifast/verifast/releases`.
  - After this, follow the steps given below:
    - tar xzf /Downloads/verifast-nightly.tar.gz
    - cd verifast-<TAB> # Press Tab to autocomplete
    - bin/vfide examples/java/termination/Stack.jarsrc # Launch the VeriFast IDE with the specified example
    - ./test.sh # Run the test suite (verifies all examples)
- **Dafny installation**
  - Installed Dafny in VS Code in ubuntu based system.
  - VS Code provides a Dafny extension, which is just a one click install, but for this we need a windows .NET framework.
  - This can be done using following command: sudo apt-get install -y dotnet-sdk-6.0
  - After this installation, Dafny will be automatically installed in VS Code.

3. (20 points) Write a summary of your project along these lines, occupying two pages. Note: I'm adding topics to the GDoc `bit.ly/CS6110-S22-Project-Suggestions` and please add details there (this is a writeable GDoc also) or even new topics are welcome to be recorded there. In any case, please profit from the ideas there. Push it into your Github which can then be used to drive your project also.
   - A project name (tentative names are OK), a topic, and about 10 lines (12pt font) on why it matters to someone studying SW verification.
   - A description of the verification technologies that you'll learn by doing this project. (It could be a topic not yet covered in class such as static analysis.)
   - A few diagrams and other details you like to add to give me a better idea of the work.
   - If you'd like to have a project partner, plz note that.
   - Assuming you have 70% of the CS 6110 time available for your project, a brief timeline of how you'll deliver your working project by the first day of the Spring exam week.

(a) **Project**

- **Title:** Proving C programs for network flow problems (like Ford–Fulkerson algorithm) using frama-c. I have already installed Frama-C on my system.
- Network flow algorithms are graph based algorithms and each edge has some numerical capacities. The goal is to create a flow, such that the constraint on each edge is maintained.

(b) **Why frama-c**

- Most codes are still written in C language and verification of these programs are important. Learning and using frama-c helps software verification people to have the ability to do verification of different c programs by using static analysis.
- Static analysis is done using Hoare's logic which is a very essential part of software verification.

(c) **Project partner**

- No project partner yet. If you think I should have a project partner, I am ready to include one.

(d) **Basic Timeline**

- **Week 1 (March 21 -26) :** Learning Frama-C, trying to implement the codes already available. Making the basics of Hoare's logic stronger.
- **Week 2 (March 27 - April 2) :** Writing the C code for network flow algorithm in Frama-C. Also, keep learning Frama-C along side.
- **Week 3 (April 3 - April 9) :** Writing other codes for network flow analysis in Frama -C.
- **Week 4 (April 10- April 17) :** Can I do a bigger project using the same technique and same direction. (A discussion with professor for the same)
- **Week 5 (April 18 - April 23) :** Wrapping up
- **Week 6 (April 24 - April 28) :** Project report and presentation.

(e) This is a very rough idea of what I am planning to do. I would really like to take your input and suggestions. Also, I will update my proposal and work as I keep going in the direction.