Technische Universität Braunschweig

Institut für Softwaretechnik und Fahrzeuginformatik

User Study

Exercise Sheet A

11th February 2021

1 General Information

This user study consists of two tasks to solve, one with the help of WebCorC and KeY. Please refrain from using any external libraries. You should only use **while-loops** in the Java programming tasks. At the end of this document, there is a cheat sheet with some syntax help.

WebCorC can be accessed via the address:
http://134.169.29.219:8080/de.tu_bs.cs.isf.cbc.web/

When you start the task with WebCorC, open the link and you will see the WebCorC interface. Before starting the modeling, make sure to note down your session-id (for example temporarily in a .txt-file) which is displayed at the top right. You can copy the ID with a mouse click. If no ID is shown, wait some seconds and refresh the page until you see your ID. It is needed to map the questionnaire to your tasks later on. Once the start signal is given, please begin with the first task. You will have 30 minutes for each task. If you finish earlier, you can move on to the next task. After the tasks are processed, answer the questionnaire that comes within the file package.

2 First Task - MinSearch

Open the folder *Exercise1*. You should find the file for a *MinSearch* algorithm. You are given a specification and are not allowed to change it. The first task describes an algorithm that finds the smallest element in an array. The array is traversed and the index of the smallest element is tracked. The global conditions ensure that the array has at least one element.

Solve this task using **WebCorC**. For this, you have to import the JSON-file into WebCorc by clicking "File -> Import Graph" top left in the navigation bar and choosing the appropriate file. If you need to reload the file, you need to refresh the website in beforehand.

The WebCorC model contains a formula with the specification. Use the refinements on the left to create your solution. It is possible to zoom out a bit by holding strg + scrolling, if the elements are shown too big. You can connect them by drag and drop on the "pins" at the edge of the element box. This will also propagate the conditions. To verify your refinements, use a right click anywhere and choose "Verify All". For now, it is only possible to verify all statements at once, which might take some seconds. If they were verified successfully, they change their color to green, else they will change to red.

3 Second Task - Modulo

Open the folder *Exercise2*. There, you will find the file for a *Modulo* algorithm. You are given a specification and are not allowed to change it. Your task is to describe an algorithm that determines the quotient and the remainder of a division such that the formula $a = b * factor + remainder$ is fulfilled. You are **not** allowed to use the operator / and % in this task. a and b are limited to natural numbers bigger or equal zero (a), and bigger zero (b). They also have an upper limit to decrease verification time.

Solve this task using **Java programming**. If necessary, specify loop-invariants. You can code in Eclipse or similar, but to verify your Java code, please upload your .java file with the button "Upload Java File" on WebCorC. You will receive a response whether the proof could be closed or not. For now, it is necessary to reload the website before your next verification, else the tool won't trigger a new verification process.

Important: Calculate the values of *factor* and *remainder* with local variables and assign their values to the class variables at the end after the calculation.

4 Questionnaire

Nearly done! For the last part, please open the *Questionnaire.docx* and take your time answering the questions. Remember to assign your session-id to the questionnaire and hand it in with the instructions inside.

5 Cheat Sheet

Some practical hints:

Java Modeling Language

- Loop-invariant:

  ```
  /*@ loop_invariant a > b;
  @ decreases a;
  @*/
  ```

- And: &&
- Or: ||
- Implication: ==>
- Bi-Implication: <==>
- Comparing two values: ==
- Universal quantifier: (\forall int a; a > 0; a > b);
- Existential quantifier: (\exists int a; a > 0; a > b);

WebCorC

- And: &
- Or: |
- Implication: ->
- Bi-Implication: <->
- Comparing two values: =
- Universal quantifier: (\forall int a; ((a > 0) -> (a > b)));
- Existential quantifier: (\exists int a; ((a > 0) & (a > b)));
- Global Conditions: These conditions will be added to all proofs as pre- and post-condition
- Variables: Define all your variables that you use here
- Take note that conditions don't end with a semi-colon, whereas statements do – WebCorC doesn't inform you about wrong syntax for now