Critical Systems (CSCM13)
Coursework

Anton Setzer

14 February 2020

General Remark:

- **Deadline: Monday, 27 April 2020, 11:00 AM**.
- There are **different coursework** for CSCM13 and CSC313. Please make sure that you attempt the **appropriate coursework.**
- All code needs to **compile/pass SPARK Ada using the 2015 version** (which is the version installed in the Linux lab).
- You need to submit as one zip file, electronically the following:
  - The source code of the final program in questions 1 - 3.
  - The output of the why3 system in question 1 (h) (this should only contain the most relevant part of why3 similarly as it was done in the lecture, without any parts which have as goal a range condition).
  - One pdf file containing the required documentation for Question 3.

  The format of your solution should be as follows:

  - Only a zip file (especially **no rar** file) is allowed.
  - **Word Documents are not allowed**.
  - The documentation for Question 3 should be a **single** pdf file.
  - Subdirectories should **not** be in zipped form, the only zipped part should be the file you submit.
  - Your files should have a clear directory/folder structure (e.g. pdf, question-1, question-2, question-3, with possible subdirectories such as question-1-f question-1-h etc)
  - You should use meaningful filenames (not e.g. example, or names inherited from the examples given to you),
  - It should be possible to check using Ada/SPARK Ada each subdirectory/folder of the source code directly without having to add any files such as main.gpr or library files and contain only the .ads .adb files needed for this particular question.
  - The solution needs to be submitted **via Blackboard**.
  - As a backup send as well your solution as an email to `cssubmissions@swansea.ac.uk`.
    - Note that this is an unmonitored mailbox.
    - It serves only as a backup, so that if something goes wrong with the blackboard submission a backup copy can be retrieved from there.

    * In case **you know that you fail to submit via blackboard**, you should send a copy of your coursework to `a.g.setzer@swansea.ac.uk`, where it is easier to retrieve your solution. This should only happen in exceptional cases.
    * If a solution is only submitted via email, we reserve the right to deduct marks from your solution.

- Failure to follow the instructions above will result in marks being deducted.
- Subquestions which you have been completed during the lab sessions will be signed off during the lab session. However, you need to sign off only the latest subquestion you have done in question 1 and 2.
  - But any marks will be **subject to submission of printouts and the source code** as mentioned before.
- All conditions added (data flow, information flow, verification conditions) are only accepted, if they **express the correctness of the intended program**, as far as it can be expressed by the conditions.
- Statement regarding **Academic Integrity**:

  By submitting this coursework, electronically and/or hard copy, you state that you fully understand and are complying with the university's policy on Academic Integrity and Academic Misconduct. The policy can be found at `https://www.swansea.ac.uk/academic-services/academic-guide/assessment-issues/academic-integrity-academic-misconduct`.

Question 1
Consider the following fragment of code of a function with input X and Y:

```
Aux := X + Y;
if Aux < 360 then return Aux;
else return Aux - 360;
end if;
```

This code accepts two angles which are supposed to be integers between 0 and 359, adds them and computes the resulting angle, which is obtained by possibly deducting 360 in order to be in the range of angles.

(a) Define a procedure which computes the sum of two angles as specified above. [3 marks]

(b) Add a main procedure which asks the user for 2 angles, runs the procedure from (a), and displays the result obtained by applying the procedure. The main procedure should run in a loop, which at the end asks whether the user wants to continue with another example or wants to terminate. It should only accept user inputs which are in the range from 0 to 360.

If you follow the example in the example collections
misc/conversion24hrClock12hrAMPMClock/vers2OnlyAdaWithIOIntegersNoLibrary/
your program will raise an exception, if the user instead of an integer inputs something else. You can ignore this problem at this stage. When using SPARK Ada later, we will use a library, which takes care of this problem.

[3 marks]

(c) Adapt the solution from (b) so that numbers make use of a type of angles in the range from 0 to 359.

[3 marks]

(d) Adapt the solution from (c) by adding a function, which has the same functionality as the procedure, but which takes as inputs the two inputs and returns the result. The main procedure should allow to execute the function and the procedure.

[3 marks]

(e) Add to your solution from (d) a depends clauses to the procedure, and replace (if you haven't done so yet) the use of Ada.Text_Io and Ada.Integer_Text_IO by using the aslibrary available in the lecture example in asLibraryIO (the files were included as well in most of the examples in misc/conversion24hrClock12hrAMPMClock/) Make sure that your code passes the data flow and information flow analysis of Spark Ada.

[4 marks]

(f) Add verification conditions to the procedure and function of (e) expressing the correctness. The verification conditions should express that if one converts the inputs and the output into integers, the output is the sum of the inputs, or the sum of the inputs reduced by 360. Show that your program passes the verification conditions of SPARK Ada, including any range conditions.

[5 marks]

(g) View the generated verification conditions from (f) for the procedure (not the main procedure and not the function) using the why3 system. Create a text file containing the output of the part of the why3 system the essential generated verification conditions, which show that from any preconditions follows the post condition. You can ignore any verification conditions which have as goal that a variable is within a range. [4 marks]

Question 2

Consider the following fragment of code:

```
I:= 0;
Res := 0;
loop
  I := I + 1;
  Res := Res + K;
  exit when I = N;
end loop;
```

Assume N is > 0. This code iterates I starting from 0 to N and in each step adds K to Res. At the end I contains N and Res contains $N * K$, so it multiplies N and K.
Note that is is not a program you would usually write (you would simply write I:= N;
Res := N * K). However this little piece of code is well suitable to explore basic features of SPARK Ada.

(a) Write in Ada one procedure which has parameters N, K, I, Res and executes this little piece of code.

[4 marks]

(b) Add files main.adb and main.ads defining a procedure which asks the user for the values of N and K, and returns the values of I and Res returned by the procedure.

[4 marks]

(c) Add depends clauses so that your program passes SPARK Ada's data and information flow analysis.

[5 marks]

(d) Add suitable pre- and post-conditions and intermediate conditions to your procedure and verify using SPARK Ada that your program is correct w.r.t. these conditions. In this subquestion you can ignore failed overflow checks. In order to check it you can use the file mainWithoutRangeCheck.gpr instead of main.gpr which is available in your git repository in `~/git/criticalhighintegritysystems/lib/mainWithoutRangeCheck.gpr`
This file switches checking for range conditions for integers off (but it keeps range conditions for subtypes such as the type of angles).

Your pre- and post-condition should express that your program is correct, i.t. that at end of this code N = I and Res = N * K.

[7 marks]

(e) Add additional verification conditions so that there are no longer any errors regarding overflows. This can be obtained by adding conditions that the variables are in suitable ranges, e.g. that N is in the range 0 .. 1000. In order to pass the verification conditions, you will need to add similar conditions at assert statements, loop invariants and possibly other places in your code.

[5 marks]

Question 3

The goal of this question is to develop a small example of a safety critical system.
Choose a simplified example of a critical system (e.g. a simple control of a rocket or of a nuclear power station, the control of an extremely simple railway system, very simple control of air traffic). You are not expected to understand the technical details of such system apart from what is common sense, and you can make some reasonable assumptions about its behaviour. For instance, when controlling a rocket, you might demand that the temperature and pressure inside must be between certain values (which you can choose using common sense), that the propulsion is sufficiently strong, etc.
The interface should be simple console input/output. For instance the program might ask for the current temperature, pressure, and then determine an action (like opening of some valve, self-destruction of the rocket etc.). Your program should consist of one loop in which the user is asked for suitable parameters and the results calculated by the system are presented.
The emphasis in this project is not on writing a complicated program with an interesting user interface (restrict yourself to console input and output and a rather simple behaviour, but in demonstrate that you understand how to verify such a program using SPARK Ada.
However some marks are reserved in the last subquestion for really clever solutions. Such solutions need not be long, but have some more interesting aspects to it.

(a) Specify your program taking into account that it is a safety critical system. About 1 - 2 pages are sufficient. [7 marks]

(b) Carry out a hazard analysis of your system using one of the techniques taught in the lecture [8 marks]

(c) Write your program so that it compiles with the compiler supplied as part of the SPARK Ada package. In your printouts include examples of runs of your system, which demonstrates the main features specified in your system. [6 marks]

(d) Add depends clauses so that your program passes SPARK Ada's data and information flow analysis. [4 marks]

(e) Add suitable pre- and post-conditions and verify using SPARK Ada that your program is correct w.r.t. these conditions. [10 marks]

(f) The marks in this subquestion are reserved for really good and deep solutions which go beyond. Such solutions need not be very long, but they should have some interesting aspects in it.

[15 marks]