University of Toronto - Department of Computer Science
CSC410, Fall 2016
Assignment 1

This assignment is worth 15% of your final mark
Due: Sunday, October 23, 2016 at 23:59

Instructions

The assignment is to be done in groups of two. Complete the provided files with appropriate annotations (do not change the file names and the implementation unless you're specifically asked to in the questions) and submit them to Markus.

Questions? Ask them on Piazza (folder 'assignment1').

Both of the group members should sign this document.
==
CSC410, Fall 2016 - Assignment 1
Name: _______________
Student Number: _______________
Lecture: o Monday   o Tuesday

Name: _______________
Student Number: _______________
Lecture: o Monday   o Tuesday

We are the sole author of this homework.   Signatures: _______________
==

Using ESC/Java2 and Dafny

We will be using ESC/Java2 and Dafny for this assignment. They are available on the CDF machines. Usage:

- ESC/Java2 can be directly run from command line:
```
werewolf:~% escjava2 QuickSort.java
```

- Dafny has to be run with 'mono' (pre-installed on CDF):
```
werewolf:~% mono /u/csc410h/fall/pub/dafny/Dafny.exe prisoner.dfy
```

This will also compile an executable for the input file: `prisoner.exe`
To make your life easier, add an alias in your '.bashrc' ('.cshrc' depending on the shell you use):

```
alias dafny='mono /h/u2/csc410h/fall/pub/dafny/Dafny.exe'
```

or in .cshrc

```
alias dafny mono /h/u2/csc410h/fall/pub/dafny/Dafny.exe
```

Then you can run Dafny by:

```
werewolf:~% dafny prisoner.dfy
```

Dafny also has a web interface: http://rise4fun.com/Dafny

Problem 1: `OpenInterval [30 marks]`

Consider the file `IntervalDriver.java` (provided along with this document). There are two classes defined, i.e., `IntervalDriver` and `OpenInterval`. Both classes have been annotated with some method contracts (maybe incomplete) and class invariants. An `OpenInterval` represents an interval with open end points (non-inclusive), i.e., (low, high), where 'low' represents the lower bound and 'high' represents the upper bound. The `joinIntervals()` method joins two intervals if they are overlapping, or returns an empty interval otherwise.

For example,

(2,3) joins (2,4) gives (2,4),
(2,4) joins (4,5) gives empty interval,
(2,4) joins (5,7) gives empty interval,
(1,3) joins (2,4) gives (1,4).

(a) Add appropriate contracts for the two constructors in the `OpenInterval` class. [5 marks]
(b) Add appropriate method contracts for `getLow()` and `getHigh()`. [5 marks]
(c) Complete the method contracts for the method `equals(OpenInterval)`. [5 marks]
(d) There might be a bug (or two) in the method `joinIntervals(OpenInterval, OpenInterval)`. Check the program using ESC/Java2 and use the warning messages to diagnose the problem(s). Fix the bug(s) without changing the method's intention, i.e., the returned interval should only be empty when the input intervals are non-overlapping. Add sufficient JML annotations to remove all ESC/Java2 warnings. You are not allowed to change or add annotations in the `IntervalDriver` class. You are not allowed to use `assume` and `nowarn`. [15 marks]

Problem 2: QuickSort [60 marks]

Consider the file `QuickSort.java` (provided along with this document).

(a) Write the method contract for the function `sort()`. Check `QuickSort.java` with ESC/Java2. A good method contract should produce the following warnings. [15 marks]:

```
Quicksort.java:14: Warning: Postcondition possibly not established (Post)
  }
  ^
Associated declaration is "Quicksort.java", line 9, col 6:
      ensures (\forall int i; (0 < i && i < a.length) ==> a[i-1] <= a[ ...
      ^
Quicksort.java:21: Warning: Possible null dereference (Null)
          quicksort(a, start, p, a[p], llimit);
                                  ^
… …

Quicksort.java:55: Warning: Possible negative array index (IndexNegative)
      a[i] = a[j];
              ^
Quicksort.java:55: Warning: Array index possibly too large (IndexTooBig)
      a[i] = a[j];
              ^
16 warnings
```

(b) Add specifications to the other functions so that all warnings are eliminated. Do not change the given implementations. Note that when you introduce certain specifications, new warnings will appear. You are not allowed to use `assume` and `nowarn`. [45 marks]

Problem 3: The Prisoner of Benda [60 marks]

Professor Farnsworth's mind-switching machine has been used unwittingly by the crew to switch minds. Unfortunately, the machine can switch minds between any pair of bodies only once. Can everyone be switched back if we use more bodies? Oh dear, I am afraid we need to use ... math! Good news everyone! The Harlem Globetrotters have devised a method to switch everyone back using only two extra bodies. The problem and algorithm are outlined at: https://en.wikipedia.org/wiki/The_Prisoner_of_Benda. You may also find this video helpful for understanding: https://www.youtube.com/watch?v=J65GNFfL94c. Let's get on with it then.

Consider the file `prisoner.dfy` (provided along with this document). The array `L` represents a permutation of the minds of the crew, where index `i` is body `i` and contains mind `L[i]`. Assume that minds and bodies are numbered `0` through `L.Length` making `L` a permutation of `0,..L.Length`. So originally body `i` had mind `i`.

It turns out only two extra bodies are required. You have been provided with code that solves the problem and switches everyone back, including the two extras. You can test the program by compiling it with Dafny and run the executable: `./prisoner.exe`

Write appropriate pre-/post-conditions and invariants to solve and verify this problem using Dafny. You must not modify the code, and you must not modify the contracts (pre-/post-conditions and invariants) of the benda method.

(a) Add appropriate loop invariants for the while-loop in the `benda` method. [20 marks]
(b) Add appropriate pre-conditions for the `cycle` helper method. [15 marks]
(c) Add appropriate post-conditions for the `cycle` helper method. [15 marks]
(d) For full marks, your code must not have any errors when verified by Dafny. Fix any remaining warnings by providing the appropriate annotations. [10 marks]