Université d'Ottawa
Faculté de génie

École de science
d'informatique
et de génie électrique

uOttawa

L'Université canadienne
Canada's university

University of Ottawa
Faculty of Engineering

School of Electrical
Engineering
and Computer Science

Assignment 0
CSI2120 Programming Paradigms

Winter 2019

Part 2 and 3 due on March 15th before 11:00 pm in Virtual Campus

[6 marks]

Problem Description

See the separate problem description first.

Part 2: Concurrent solution (Go) **[3 marks]**

The next step in the comprehensive assignment is to code the Stepping Stone method in Go. The main difference in your Go solution compared to the Java solution is that classes are replaced by structs and methods. You are free to change the structure of your program, i.e., your Go program does not have to follow the design of your Java program.

You are allowed to use the initial solution as created by your Java program (or by other means, e.g., manually or with another solver). Your Go program should ask the user to specify a text file containing the description of the problem (same as before for your Java code) and, different than before, for a text file containing an initial solution. Your Go program should then find an optimal solution (table including minimal cost) and write it to a text file named solution.txt

In addition, your program must use Go routines to find the solution using concurrency. You must create a function to get the marginal cost of a path from an empty cell.

marginalCost (cell Cell, result channel Path)

This will allow you to run the marginalCost function concurrently for all empty cells of the current iteration using the go keyword. Next, your program identifies the path and performs a new iteration.

You must submit your Go source code well commented, in which all types and methods created are clearly described.

Part 3: Logic solution (Prolog) **[3 marks]**

The next step in the comprehensive assignment is to code the Stepping Stone method in Prolog. It seems natural to approach the problem with a generate-and-test approach. However, you are free to choose the structure of your program, i.e., your Prolog program does not have to use a generate-and-test approach nor does it need to follow the design of your Java or Go program.

For Prolog, you are again allowed to use the initial solution as created by your Java program (or by other means, e.g., manually or with another solver). Your predicate should expect as input two file names: One filename of a text file containing the description of the problem and a second text file containing the initial solution (same as before for your Go code). Your Prolog program should then find an optimal solution (table including minimal cost) and write it to a text file named solution.txt. Please name your top-level predicate:

```
minimumTransportCost('input.txt', 'initial.txt', Cost)
```

This predicate will need to accept two file names and will be true if Cost is minimal for the specified problem. Note that Cost is intended as output.

You must submit your Prolog source code well commented. Each helper predicate has to have a clear description when it is true.

Note that a file to list reader is available from the general problem description page.