CS 9868 Internet Algorithmics
Assignment 2: Graduate and Undergraduate Students
Due October 26 at 11:59 pm

This assignment has 10 bonus marks. For each one of the following questions you need to use the simulator for synchronous distributed algorithms written by Daniel Servos (available at cs1.ca/simv2). **You need to use version 2.1.0 of the simulator**. If you are not sure what version of the simulator you have, please download an updated version from the above link. You need to submit through OWL the java files implementing your algorithms and a document (preferably in pdf format) containing your answers. No hard copy of your answers will be required this time.
If you do not have any programming experience and are not able to write java code, we will allow you to submit your algorithms in detailed pseudocode similar to the one used in class. However, this should be the exception, not the rule; you are strongly encouraged to write your algorithms in java and test them using the simulator.
In all problems below it is assumed that each processor has a unique identifier. Also, each processor knows who its neighbours are, but it does not know the number of processors in the network. Whenever you are asked to compute the time complexity or communication complexity of an algorithm you must explain how you computed the complexities and you must also give the order of the complexities.

1. Design and implement in java a distributed synchronous algorithm for counting the number of processors in a network with a ring topology. The algorithm must output the number of processors in the network. You can assume that each processor knows who its left neighbour is and who its right neighbour is. You cannot assume that a processor has been selected as the leader.

   - (5 marks) Give an informal, high level description of the algorithm in English.
   - (15 marks) Submit a java implementation of your algorithm.
   - (5 marks) Compute the time complexity and communication complexity of your algorithm.

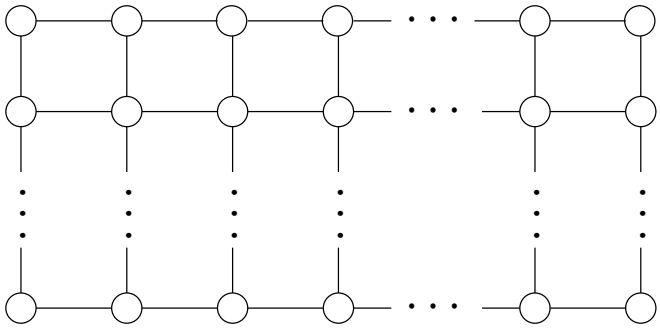
2. Design and implement in java a synchronous distributed algorithm for solving the leader election problem, but this time assume that the processors are connected in a line as shown in the figure below. The algorithm must output either "leader" or "not leader" depending on whether the processor executing the algorithm has the largest id in the network.

   Assume that each processor knows its right neighbour (if any) and its left neighbour (if any). Please read the notes at the end of the assignment to learn how to specify the configuration file for the simulator and how a processor can determine whether it has only a right neighbour or only a left neighbour.

   - (5 marks) Give an informal, high level description of the algorithm in English.
   - (15 marks) Submit a java implementation of your algorithm.
   - (5 marks) Prove that your algorithm terminates.
   - (5 marks) Prove that your algorithm produces the correct output.
   - (5 marks) Compute the time complexity and communication complexity of your algorithm.

3. Design and implement in java an algorithm for solving the leader election problem on a mesh. A network with a mesh topology consists of a set of processors arranged in a two dimensional grid, as shown in the following figure. The algorithm must output either "leader" or "not leader" depending whether the processor executing the algorithm has the largest id in the network.

   Assume that each processor knows its neighbours and their relative positions in the grid, i.e it know the neighbour on its left (if any), the one on its right (if any), the one on top (if any), and the one below (if any). Processors do not know the size of the mesh.



- (5 marks) Give an informal, high level description of the algorithm in English.
- (25 marks) Submit a java implementation of your algorithm.
- (5 marks) Prove that your algorithm terminates.
- (7 marks) Prove that your algorithm produces the correct output.
- (8 marks) Compute the time complexity and communication complexity of your algorithm. For the analysis of the complexities, denote with W the width of the mesh (number of processors in one column of the grid) and with L its length (the number of processors in one row of the grid).

(Rounds)time:L+W; LW=(L-1)W +

Simulator Notes

Make it sure that the first line of the network configuration file is

```
Mode Identical
```

For the second question, add the following line as the second line of the network configuration file so the simulator correctly displays a network with a line topology:

```
GUILayout line
```

In the configuration file the identifier 0 cannot be assigned to any processor, as 0 has a special meaning in the simulator. When specifying the set of neighbours of a processor, the list of neighbours in the `AddNode` command can include one or more times the identifier 0 to indicate the lack of a neighbour. For example if the network configuration file for a network with a line topology contains the command

```
AddNode 23:  0 54
```

This means that the processor with id 23 does not have a left neighbour and its right neighbour has id 54. You can use the following java code to get the neighbours of a processor and to determine whether it is the leftmost processor or the rightmost processor.

```
Vector<String> v = neighbours();
String leftNeighbour = (String) v.elementAt(0);
String rightNeighbour = (String) v.elementAt(1);
```

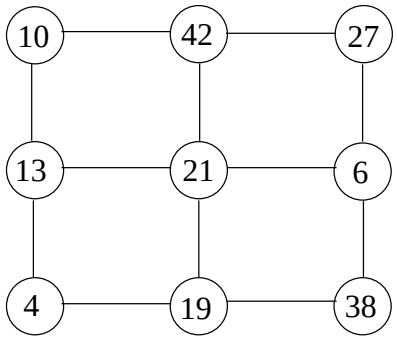
```
boolean leftmostProcessor, rightmostProcessor;
if (equal(leftNeighbour,"0")) leftmostProcessor = true;
else leftmostProcessor = false;
if (equal(rightNeighbour,"0")) rightmostProcessor = true;
else rightmostProcessor = false;
```

For a network with a ring topology you can use the first 3 lines of the above java code to determine the left and right neighbours of a processor.
In a network with a mesh topology, a processor can have 2, 3, or 4 neighbours. When specifying the neighbours of a node in an `AddNode` command, give first the id of neighbour that is to the left of the current processor (or 0 if no neighbour appears to the left), then the id of the neighbour that is above (or 0), then the id of the one on the right (or 0), and finally, the id of the neighbour below (or 0). For example the following fragment of the network configuration file:

```
AddNode 4:   0 13 19 0
AddNode 13:   0 10 21 4
AddNode 10:   0 0 42 13
AddNode 19:   4 21 38 0
AddNode 21:   13 42 6 19
AddNode 42:   10 0 27 21
AddNode 38:   19 6 0 0
AddNode 6:   21 27 0 38
AddNode 27:   42 0 0 6
```

3 col: R
4 col: D
2 col: U
1 col: L

represents the following mesh network.



Keep in mind that the algorithm used by the simulator to render the network on the screen might not show the network as the above figure. If the simulator does not draw the nodes in the position that you expect, you will have to manually re-arrange them so the network is displayed as you wish.