CSCI 5311/4311 – Computer Networks

Programming Assignment #1

Due: Hard-copy, Thursday, April 7th, 2016 by 6:00 PM, in class.
Due: Soft-copy, Thursday, April 7th, 2016 by 4:00 PM, via moodle.

Instructions:

- Please carefully review "integrity" section from the 'syllabus page' before starting the assignment. The assignment must be done by you alone.
- **5% overall bonus will be given** where 100% accuracy is achieved for all test cases (checked by the instructor) and each round within a solution is presentation by smooth graphics.

Problem Statement:

For a given set of switches/bridges and their connections among each other, determine the spanning tree configuration for the active data-path among the switches, computed based on the spanning tree algorithm that we have learnt.

To Do:

1. [**Program-Input**] Your program will run and read the input from the input.txt file. Each line in the input file is an individual problem.

For example, the first problem given by input "5 R", indicates that there are 5 switches and the connections among themselves are randomly (R) determined (by your program).

Second line indicates problem #2. Here, a set of individual connections to be solved among 5 switches which is given by "5 1-2 1-3 1-3 2-3 3-4 …". Here the first digit "5" implies that there are 5 switches. Then, "1-2" indicates that switch#1 and switch #2 are physically connected by a wire. Note, by "1-3 1-3" it is indicated that switch #1 and switch #3 are physically connected twice: connection #1 is connecting port p2 (of switch #1) and p1 (of switch#3); whereas connection #2 is connecting p3 (of switch #1) and p2 (of switch #3). Here, p2 indicates port number #2 and so on.

The ports of a switch are to be connected/allocated from lower numbered port to higher numbered port for each connection by the order of the connection read from left to right for a given problem.

2. You program will compute the corresponding spanning tree following the algorithm discussed in the class (slides) and will draw each round of intermediate configurations including the final spanning tree configure at the end.

3. [**Program-Output**] Your outputs will print the problem configure first (such as "5 1-2 1-3 1-3 2-3 3-4 1-5 …") and then draw the computed intermediate configurations as well as the final spanning tree configuration in order – this is to be done for all the problem(s) given in the input file.

Submission:

(a) In **Report_PA1_<YourID>.doc/x**, provide all different screen outputs generated by your program for all the problems given in the input.txt.
(b) Provide the source-code and executable. The executable must be easy to run. It must not ask to install programming package and then to compile to run your code. You can code in java (see the guideline for coding below).
(c) Provide a readme file which will describe how to run your code.
(d) Compress everything in one folder except the report file and submit the compressed folder **PA1_<YourID>.zip** and the **Report_PA1_<YourID>.doc/x**.
(e) Provide the hard-copy of **Report_PA1_<YourID>.doc/x in class.**

Guideline for coding

You will **submit** all of your source code in a single java* file called SpanningTree.java. You may develop your code using separate files but you will need to concatenate your source code together into a single compilable file for submission. The file SpanningTree.java should begin with a public class SpanningTree that contains the main program which takes the input file name as a command line argument and then performs the simulation of the given switch/bridge configuration problem in the input file. The public class SpanningTree should be followed by your other classes that comprise your system. Recall that Java will only allow one class within a source file to be declared public, so your SpanningTree.java source file will look something like this:

```
public class SpanningTree {
        public static void main( String [] args ) {
        … here lies code to process the input file and solving the Spanning Tree problem …
        }
}
```

Run your program and use the input.txt to get input as:

```
$ java SpanningTree input.txt
```

* If you are using programming language other than java, please adhere to the naming convention (e.g. SpanningTree.java can be SpanningTree.c)

---- x ----