CS 3305: Data Structures
Fall 2022
Assignment 9 – Graphs – 100 points

Note: If you re-upload the files, you must re-upload **ALL** files as the system keeps the most recent uploaded submission only. No zip files!

Note 2: Never hard-code test data in the test program, unless explicitly stated in the assignment. Always allow the user to enter the test data using menu option.

Note 3: Code documentation. In addition to the author header at the top of each file, please include a comment block before each method explaining the method and what it does. Add in-line comment to explain the code within the method.

The goal for this assignment is to implement the concept of *graph reachability matrix* discussed in the slides. The assignment focuses on computing **Integer reachability matrix** as illustrated in the slides.

To keep the assignment focused, let's assume that the graph has no more than 5 nodes. This way the program would compute up to 5 matrices (denoted in the slides as A^1, A^2, A^3, A^4 and A^5).

Write a java program (named ReachabilityMatrix.java) that reads a **connected (un-weighted) graph** data from the keyboard. First it reads number of nodes in the graph (no more than 5), this input value determines the matrices size and number of matrices computed for the inputted graph. Next, it reads the actual values in the adjacency matrix of the inputted graph (that is, A^1 matrix).

To keep the reading of matrix A^1 values uniform, please read the adjacency matrix row-by-row, exactly like this: (deviation from this format results in 0 points for the assignment)

```
Enter A1[0,0]:
Enter A1[0,1]:
Enter A1[0,2]:
...
Enter A1[1,0]:
Enter A1[1,1]:
Enter A1[1,2]:
...
```

Recommendation: As an intermediate step to validate the computation of the integer matrices, test your code using the example graph given on slides 23-24. Printout A^1, A^2, and A^3 for that graph and compare your outputs to the matrices shown on slide 24. Remove this testing code from the final submission.

Develop 10 **separate methods** properly names to produce each of the following output:

1. Print out the input matrix.
2. Compute and print out the graph reachability matrix.
3. Compute and print out the In-degree of each node of the graph.
4. Compute and print out the Out-degree for each node of the graph.
5. Compute and print out the total number of loops (also known as self-loops) in the graph.
6. Compute and print out the total number of cycles of length N (N is the number of nodes in the inputted graph).
7. Compute and print out the total number of paths of length 1-edge.

8. Compute and print out the total number of paths of length N edges (N is the number of nodes in the inputted graph).
9. Compute and print out the total number of paths of length 1 to N edges (N is the number of nodes in the inputted graph).
10. Compute and print out the total number of cycles length 1 to N edges (N is the number of nodes in the inputted graph).

Include the following menu to allow the user to re-run the program for different graphs. Please DO NOT hard-code test data in your final submission. Any hard-coded data you may have used for testing must be deleted from the final submission. Selecting option 2 before option 1 should give the user an error message.

```
------MAIN MENU------
1. Enter graph data
2. Print outputs
3. Exit program

Enter option number:
```

Always re-display the menu after an option (other than option 3) is fully exercised, with blank lines before and after the menu (no menu, no points!).

For illustration and formatting purposes, selecting option 2 after entering the 3-node graph on slide 23 would result in this output (notice the output labels and blank lines). Your output must be formatted following this sample run.

```
Input Matrix:
0  1  1
1  0  0
0  1  0

Reachability Matrix:
2  3  2
2  2  1
1  2  1

In-degrees:
Node 1 in-degree is 1
Node 2 in-degree is 2
Node 3 in-degree is 1

Out-degree:
Node 1 out-degree is 2
Node 2 out-degree is 1
Node 3 out-degree is 1

Total number of self-loops: 0
Total number of cycles of length 3 edges: 3
Total number of paths of length 1 edge: 4
Total number of paths of length 3 edges: 7
Total number of paths of length 1 to 3 edges: 16
Total number of cycles of length 1 to 3 edges: 5
--------------------------------------------------
```

Additional test data is provided with the assignment in a separate file. The assignment is very specific and straight forward, mostly matrix manipulation. It must be implemented as specified. Any deviation from these requirements will not be accepted and receives no points. No exceptions.

Submission:

Do not forget to include author header in each submitted file as shown, and do not forget to document your code as stated in note 3 above. **no author header or no proper documentation, no points!**

```
// Name:          <your name>
// Class:         CS 3305/Section#
// Term:          Fall 2022
// Instructor:    Dr. Haddad
// Assignment:    9
// IDE Name:      <your IDE name>
```

Please submit your file (ReachabilityMatrix.java) to the assignment submission folder in D2L by the due date posted in D2L. Make sure that your code is running correctly right before you upload your files. **No zip files or late submissions are accepted**.