CSCI 2110 Data Structures and Algorithms
Fall 2023
Assignment No. 6

Date Given: Monday, November 27, 2023
Due: Thursday, December 7, 2023

This assignment has two programming exercises, one on hashmaps and one on graphs.

Exercise 1: The objective of this exercise is to create a simple hash map using the Java HashMap data structure. First review the module on Hashing and Hash Tables from the lectures.

Your friend is starting up a new web site that is going to have users register as members, and asks you to help implement an application that would validate a member's login.

Your task is to write a program that reads a file containing the full name, username, and password for each user. ***Create a hashmap with the username as key and the password as value. Create another hashmap with the username as key and full name as value.***

After the file is read, prompt the user to enter the login name and password. If the password is incorrect, give the user two more chances. If the password is incorrect all three times, the program quits. If the login is successful, print a welcome message. Use the first hashmap to check the username and password match, and use the second hashmap to print the welcome message.

Here's a sample input file:

```
Ichabod Crane         icrane      qwerty123
Brom Bones            bbones      pass456!
Emboar Pokemon        epokemon    password123
Rayquazza Pokemon     rpokemon    drow456
Cool Dude             cdude       gh456!32
Trend Chaser          tchaser     xpxo567!
Chuck Norris          cnorris     power332*
Drum Dude             ddude       jflajdljfped
```

In the above file, for example, Ichabod Crane is the full name, icrane is the username and qwerty123 is the password. *In practice, the passwords are hashed and encrypted, but for this program, you can store them in plaintext.*

Here are two sample runs of the program:

Sample Run 1

```
Login: rpokemon
Password: drow456

Login successful
Welcome Rayquazza Pokemon
```

Again in practice, the password is hidden when it is typed, but don't worry about that in your program.

Sample Run2

```
Login: cdude
Password: trythis

Either the username or password is incorrect. You have 2 more attempts.
```

```
Login: cdude
Password: trythat

Either the username or password is incorrect. You have 1 more attempt.
Login: cdude
Password: 675rtht!
Sorry. Incorrect login. Please contact the system administrator.
```

Exercise 2: In the lectures, we will be discussing Topological Sorting as one of the Graph Algorithms. In this assignment, you will implement this algorithm.

Write a Java program that does the following:
a) Reads an input text file like the one described below. The text file contains the description of a directed, unweighted, acyclic graph.
b) Construct an adjacency matrix representing the graph.
c) Perform topological sorting
d) Display the result of the sort.

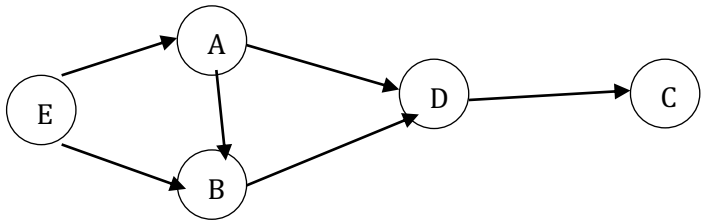
Here's the description in detail:

The input file will contain a number of lines. The first line will be an integer representing the number of vertices in a directed, unweighted graph. You can also assume that the graph is acyclic (this is a requirement for topological sorting). Each following line will contain a pair of uppercase characters (A-Z) separated by **tabs**. These pairs represent a graph's edges.

5

E	A
E	B
A	B
A	D
B	D
D	C

The sample input file describes the following graph:



After reading data from the input file, your program should create an adjacency matrix representing a graph. Declare a square, 2D array to act as an adjacency matrix. You may find the following code line useful:
int[][] adj = new int[n][n];

The sample input file results in a 5x5 matrix shown below:

```
 A B C D E
A 0 1 0 1 0
B 0 0 0 1 0
C 0 0 0 0 0
D 0 0 1 0 0
E 1 1 0 0 0
```

Both Strings and characters can be easily converted to integers in Java. A captured String can be converted to an integer using **Integer.parseInt(string)**, while the capitals from the input file can be converted to indices via a simple arithmetic expression: **character-65.** You may find the following code snippet useful when building your adjacency matrix:

```
int v0 = input.next().charAt(0)-65;
int v1 = input.nextLine().charAt(0)-65;
adj[v0][v1] = adj[v1][v0] = 1;
```

Next, implement the Topological Sorting algorithm as follows:

```
Initialize an empty queue.
for each vertex v in the graph
              compute the predecessor count, pred(v) (this is just
the indegree of the vertex)
for each vertex v in the graph
              if (pred(v)==0) add v to the queue.
topnum ←1
while queue is not empty {
w ←dequeue
assign w with topnum
topnum ← topnum+1
for each neighbour p of w{
              pred(p) ← pred(p) -1
              if (pred(p)==0) then
                add p to queue
                 }//end for
}//end while
```

All the vertices with be assigned with topnum in the topological sorting order.

Display the result. This will just be the listing of the vertices with increasing values of topnum. For the above example, one solution would be:

```
topnum:          1  2  3  4  5
                 E  A  B  D  C
```

What to submit:

ONE zip file containing the following:

For Exercise 1:
1. Source codes (.java files)
2. Sample input file (.txt file) that you used
3. Three sample runs of the program (cut and pasted screenshot into a text file).

For Exercise 2:
1. Source codes (.java files)
2. Sample input text file (.txt)
3. Sample output (cut and pasted screenshot into a text file)

You MUST SUBMIT .java file that are readable by your TAs. If you submit files that are unreadable such as .class, you will lose points. Please additionally comment out package specifiers.

Late Submission Penalty: The assignment is due on Thursday at 11.59 PM. Late submissions up to 5 hours (4.59 AM on Friday) will be accepted without penalty. After that, there will be a 10% late penalty per day on the mark obtained. For example, if you submit the assignment on Friday at 12 noon and your score is 8/10, it will be reduced to 7.2/10. Submissions past five days after the grace submission time will not be accepted.