CMPE 343
Spring 2023
Programming Homework 3

This assignment is due by 23:59 on Sunday 21, May 2023.

You are welcome to ask your HW related questions by joining the recitation hours given below:

There will be two Q&A Office Hours on the following days:

- CMPE343-HW1-OfficeHour1: May 9 Tuesday, 17:00-19:00, Zoom ID: https://tedu.zoom.us/j/96924822435
- CMPE224-HW1-OfficeHour2: May 16, 17:00-19:00, Zoom ID: https://tedu.zoom.us/j/98879381589

Note: Please make sure that you have read the HW document well before participating. However, no HW related questions will be accepted except from the above options.

PROGRAMMING TASK

In this part, you must implement your own graph data structure by taking inspiration from your textbook and use it to help to solve problem. You are not allowed to use any external library or .jar file. Any solutions without using graph data structure are not evaluated!

Question 1(25 points):

In Esenboğa airport, there is a parking facility which is in the form of a graph having *N* nodes and *M* edges. The graph does not have self-loops or multiple edges. Each node represents a parking slot and has a capacity of vehicles it can hold. Each edge has a weight of *w*, which indicates that it will cost *w* units to go from node *u* to node *v*. All parking slots have a parking fee *F* per vehicle, which is same for all slots.

There are *K* identical vehicles entering the parking facility, each vehicle enumerated with a distinct number from *1* to *K*. The vehicles enter in their natural order, that is, vehicle number *1* enters, then vehicle number *2*, then *3* and so on till vehicle number *K*. For each vehicle, you have to print the minimum total cost that is incurred on the vehicle owner. Here, total cost includes **cost of the path taken to reach the parking slot and parking fee of the slot**. It is guaranteed that you can reach any slot from any other slot. **All vehicles entering the parking facility enter from the parking slot 1.**

In the input, the first line contains 3 space separated integers, *N*, *M* and *F*. *N* denotes the number of nodes, *M* denotes the number of edges and *F* denotes the parking fee. The second line consists of *N* space separated integers denoting the seating capacity of each parking slot. Following *M* lines contain three space separated integers each: *u*, *v* and *w*, denoting we can reach from node *u* to node *v* incurring a cost of *w* units. The last line of input contains an integer *K* denoting the number of vehicles enter the parking facility.

In the first line it is given that we have 5 parking slots, 4 connections between the parking slots and 10 as parking fee which is the base fee for each car enters the parking facility. In the second line, we have 5 parking slots with their capacity which are 1, 2, 1, 1, and 2. In the next 4 lines, the distance between these parking slots is given and the number of vehicles will be entering the parking facility is 5. You need to give the total cost each car enters the facility.

Sample Input:

```
5 4 20
1 2 1 1 2
1 2 2
4 5 1
3 4 2
1 3 1
5
```

In the output, print K space separated integers denoting answer for each vehicle. ith integer in the space separated integers denotes answer for ith vehicle number. If it is not possible to enter a parking slot print −1 for that vehicle.

The output for the about inputs as follows. Please check your program with this input as well as the others that you will create. Please note that we may use other inputs when grading your assignments.

Sample Output:

```
20 21 22 22 23
```

Question 2(25 points):

There are *N* bus stations and *M* buses provide service in Ankara Kızılay. Each bus *s* has a schedule consisting of *t* stations. At the beginning, each bus at their first station and each minute it travels to the next station. When it reaches the end, it goes back to the beginning and starts again. At the beginning, you are at station 1. If at any moment you and a bus are at the same station, you can get on it and travel with it. You can get off the bus at any station. The same station can appear multiple times in the itinerary of the bus, but not adjacent to each other (in particular the last station

is adjacent to the first one). You are interested in finding the minimum amount of time you need to reach to each station, or state that it is impossible.

You can only travel in one bus at a time, but you can use multiple buses to reach your destination. Do not forget that you can only travel by bus. There can be multiple buses at the same station at the same time. Getting on and off a bus is instantaneous.

In the input, the first line contains *2* space separated integers, *N*, *M*. *N* denotes the number of stations, *M* denotes the number of buses respectively. Each of the next *M* lines contain an integer *t* followed by *t* integers which demonstrate the schedule of each bus.

In the given sample input, it is given that we have 8 stations, and 4 buses work between them in the first line. The next lines give the schedule of each bus with the number of stations it visits and which stations it visits. For example, the second bus has 3 stations 6, 1 and 2. Buses visit the stations according to the given order, so the second bus is in the 6th station at the beginning, then it goes to the 1st station and then 2nd station and returns to the 6th station. And, traveling between each station takes 1 minute, so it takes 2 minute for the second bus to go from 6th station to 2nd station.

Sample Input:

```
8 4
2 5 4
3 6 1 2
4 4 2 1 3
2 7 8
```

In the output, print *N-1* space separated integers denoting the minimum amount of time needed to reach each station except the 1st station because you are at the first station at the beginning. If any of the stations is not reachable with the given schedule of the buses, you need to print *-1* for that station. You need to print the output in numerical order which means you need to give the amount of time for 2nd station first, then 3rd, then 4th and goes on.

The output for the above inputs is as follows. Please check your program with this input as well as the others that you will create. Please note that we may use other inputs when grading your assignments.

Sample Output:

```
2 3 4 6 3 -1 -1
```

WHAT TO HAND IN

- **You need to upload your code into VPL on LMS for each question.** If you do not upload your code into VPL on LMS, your homework will **not be evaluated.**
- The Java sources should be WELL DOCUMENTED as comments, as part of your grade will be based on the level of your comments.
- You need to upload **maximum-3 pages** PDF report document that explains your own answers for programming task in a clearly readable PA report format (refer to **PA REPORT FORMAT** section).

PA REPORT FORMAT

A programming assignment report is a self-description of a programming assignment and your solution. The report must not be handwritten. You may use a word processor or the on-line editor of your choice and prepare as a PDF document. The report must be grammatically correct and use complete English sentences. Each report should include the following sections, in the order given:

Information (%2.5): This section includes your ID, name, section, assignment number information properly.

Problem Statement and Code Design (%15): Include a brief summary of the problem and/or your sub-tasks to be completed in this assignment. You should show your modular design rationale by creating a structure chart that indicates your top-down, stepwise refinement of the problem solution. You may create the structure chart using available graphical tools like MS PowerPoint, SmartDraw etc.

Implementation and Functionality (%20): Since you have modular source code, you should describe each sub-module (program) in this section. Each sub-module should include names and types of any input/output parameters as well as the pseudocode algorithm that used for completing its task. By this way, you give meaning to each chart boxes from the previous section.

Testing (%7.5): You should provide a tester class that is able to identify key test points of your program. This class should be able to generate additional (apart from the given sample input/output) test data for the purpose of being clear on what aspects of the solution are being tested with each set. This section should also include a description of any program *bugs* that is, tests which has incorrect results. You should write these to describe your tests, summarize your results, and argue that they cover all types of program behavior.

Final Assessments (%5): In this final section, you should briefly answer the following questions:

- What were the trouble points in completing this assignment?
- Which parts were the most challenging for you?
- What did you like about the assignment? What did you learn from it?

GRADING:

- Codes (%50: %25 for Q1 and %25 for Q2)
  - Available test cases evaluation on VPL: %15
  - Hidden test cases evaluation: %15
  - Approach to the problem: %20
- Report (%50: %25 for Q1 and %25 for Q2)
  - Information: %2.5
  - Problem Statement and Code design: %15
  - Implementation, Functionality: %20
  - Testing: %7.5
  - Final Assessments: %5

IMPORTANT

IMPORTANT NOTES: Do not start your homework before reading these notes!!!

1. This assignment is due by 23:59 on Sunday, May 21th.

2. You should upload your homework to Moodle before the deadline. No hardcopy submission is needed. You should upload your code files into VPL and your report.

3. The standard rules about late homework submissions apply (20 points will be deducted for each late day). Please see the course syllabus for further discussion of the late homework policy as well as academic integrity.

4. You ARE NOT ALLOWED to modify the given method names. However, if necessary, you may define additional data members and member functions.

5. Your classes' name MUST BE as shown in the homework description.

6. The submissions that do not obey these rules will not be graded.

7. To increase the efficiency of the grading process as well as the readability of your code, you have to follow the following instructions about the format and general layout of your program.

8. Do not forget to write down your id, name, section, assignment number or any other information relevant to your program in the beginning of your Java files. Example:

```
//-------------------------------------------------------
// Title: Scheduler tester class
// Author: Name/Surname
// ID: 2100000000
// Section: 1
// Assignment: 1
// Description: This class tests the …
//-------------------------------------------------------
```

9. Since your codes will be checked without your observation, you should report everything about your implementation. Add detailed comments to your classes, functions, declarations etc. Make sure that you explain each function in the beginning of your function structure. Example:

```
    void setVariable(char varName, int varValue)
//-------------------------------------------------------
// Summary: Assigns a value to the variable whose
// name is given.
// Precondition: varName is a char and varValue is an
// integer
// Postcondition: The value of the variable is set.
//-------------------------------------------------------
{
                // Body of the function
  }
```

10. Indentation, indentation, indentation...

11. This homework will be graded by your TAs, Deniz Merve Gündüz, Elif Ünal and Enes Arslan. Thus, you may ask them your homework related questions through HW forum on Moodle course page. You are also welcome to ask your course instructors Tolga Kurtuluş Çapın and Ulaş Güleç for help.