CMPE 343
Spring 2022
Programming Homework 1

This assignment is due by 23:55 on Friday 23, March 2022.

You are welcome to ask your HW related questions. You should use only one of these options:

1. Moodle Homework **Question Forum**: HW Question-and-Answer (Q&A) Forum on Moodle is always available. Use the "Forum" link at the course Moodle page.
2. Homework **RECITATION HOURS**: There will be two Q&A RECITATION HOURs on the following days:
   - CMPE343-HW1-OfficeHour1: March 11, 06:00-07:00 PM, Zoom ID: https://tedu.zoom.us/j/98929958529
   - CMPE224-HW1-OfficeHour2: March 18, 06:00-07:00 PM, Zoom ID: https://tedu.zoom.us/j/99286862384

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.

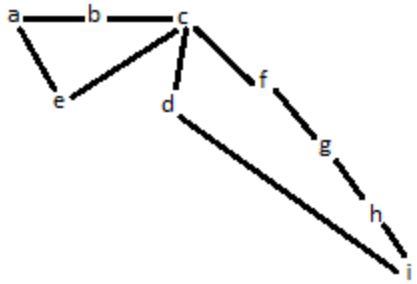
PROGRAMMING TASK

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!

You are a famous civil engineer, and you want to make your country better by collecting some information about roads between cities.

Question 1(25 points):

In this question, you are expected to find all cycles that contains specified city. In this way, you can arrange roads for a given city. Following example explains expected output.



Let assume that we want to find all cycles that contains city C. The first cycle is c-e-b-a and the second cycle is c-f-g-h-e-d. While printing these cycles, you should arrange cities in each cycle alphabetically, e.g The first cycle should be printed as a-b-c-e. Also, if you find more than one cycles, you should print based on number of cities in ascending order for each cycle. Therefore, final output should be as follows for above Figure:

1) a-b-c-e
2) c-d-f-g-h-i

If you can not find any cycles that contain given city, just print "**No cycle**".

City and road information should be read from the user.

Here is the `sample argument`:

```
a-b,b-c,a-e,c-d,c-f,f-g,g-h,h-i,i-d  //First, city-road information
c                                    // Second, read city name
```

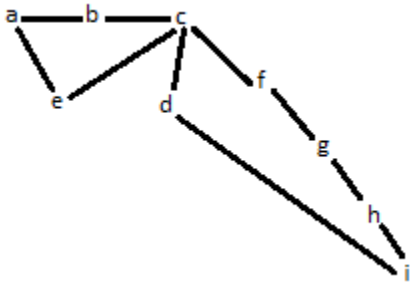
Here, a-b means that there is a path between cities a and b and vice-versa.

The output for the above input 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 input when grading your assignments.

```
1) a-b-c-e
2) c-d-f-g-h-i
```

Question 2(25 points):

In this question. you are expected to find a path between given two cities that contain given number of cities.



For above cities and roads, lets assume that we want to find a path between cities a and i. Also, this path should have exactly 5 cities. Therefore, there are two paths that satisfy these conditions: which are **a-b-c-d-i** and **a-e-c-d-i** . While printing these paths, you should arrange cities in each path alphabetically, e.g The first path should be printed as a-b-c-d-i and the second path should be printed as a-c-d-e-i. Also, if you find more than one cycle, you should print path lexicographically. Therefore, final output should be as follows for above Figure:

1) a-b-c-d-i
2) a-c-d-e-i

If you can not find any path, just print "**No path**".

City and road information should be read from the user.

Here is the `sample argument`:

```
a-b,b-c,a-e,c-d,c-f,f-g,g-h,h-i,i-d  //First, city-road information
a                                    // Second, read first city name
i                                    // Third, read first city name
5                                    // Fourth, # of cities
```

Here, a-b means that there is a path between cities a and b and vice-versa.

The output for the above input 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 input when grading your assignments.

1) a-b-c-d-i
2) a-c-d-e-i

WHAT TO HAND IN

A zip file for both parts containing:

- The Java sources for your program.
- The Java sources should be WELL DOCUMENTED as comments, as part of your grade will be based on the level of your comments.
- You can test your Java source files on available Moodle VPL environment to ensure your solution's correctness before submitting. VPL simply tests your program's output by checking against given sample input.
- A **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 hand-written. 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, Functionality, Performance Comparison (%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. Also, you should add your performance comparison, part II, here.

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 23:55 on Wednesday, March 23 th.

2. You should upload your homework to Moodle before the deadline. No hardcopy submission is needed. You should upload files and any additional files if you wrote additional classes in your solution as a single archive file (e.g., zip, rar).

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, Bedrettin Çetinkaya, Deniz Merve Gündüz. 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 Çapın for help.