CMPE 343
Fall 2020
Programming Homework #3

This assignment is due by 23:55 on Wednesday, 2 December 2020.

Late submissions: 20 points will be deducted for each late day.

PART I (60 points)

Heroes live in a big fantasy world with *N* cities. Since they have lots of job to handle around many years, they agreed to use special passages for transportation. Every passage must be one-directional. Heroes have already decided which passages they want to use and in which order they should be.

Head of the heroes, Jason Scott, asked for your help to check the list of passages and find which of them should not build. He doesn't expect that heroes get stuck in an infinite loop with passages since their time is worthy.

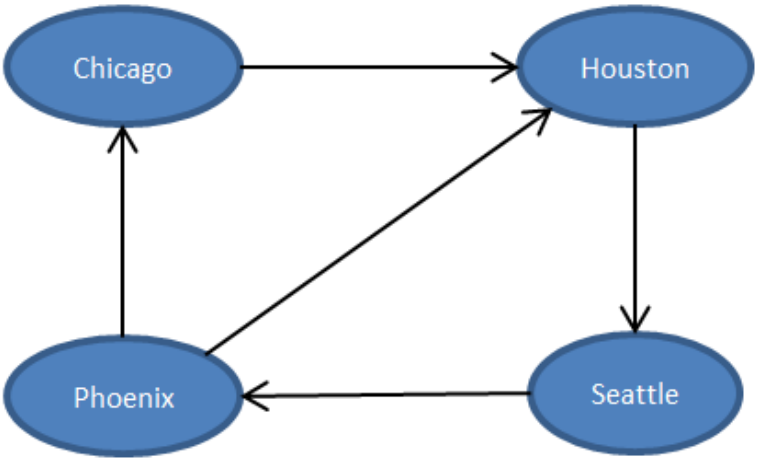
Input file is a txt file that includes the information to build your directed graph structure as follows:

- First line shows number of cities, *N*, in fantasy world (cities are labeled between 1 to *N*)
- Next line represents number of passages, *T*
- Remaining next *T* lines are the decided passages by heroes (i.e. one-directional directed edges between cities)

Here is the sample input txt file example:

```
4
5
Houston Seattle
Seattle Phoenix
Phoenix Houston
Chicago Houston
Phoenix Chicago
```

And for illustration purpose, the respective Diagraph for above input file is:



Since it is a Diagraph, you should build the edges according to the order they appear in input file. As a hint, you can think this problem as a DFS application on directed graphs to detect loops. If the current edge does not form a cycle with other edges already constructed, we build it; otherwise, we inform that edge and ignore it. You may use the Digraph and related classes in our textbook and its web site (https://algs4.cs.princeton.edu/code/edu/princeton/cs/algs4/Digraph.java.html).

Your final output must print the passages that should not build on this world and it should be ended with a line "0 0" as follows:

```
% java MyMain sampleinput1.txt

Phoenix Houston
Phoenix Chicago
0 0
```

PART II (40 points)

Jason Scott has *M* jobs to handle. However, these jobs are not independent and the execution of one job is only possible if other jobs have already been done. He asked for your help to determine a possible order of getting all the jobs done.

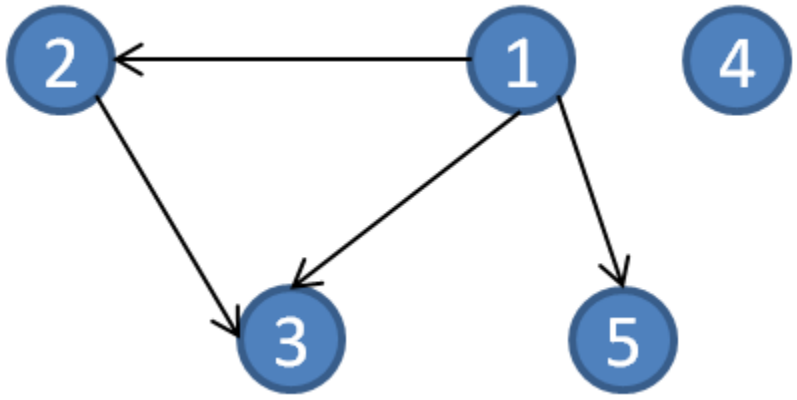
Input file is a txt file that includes the information to build your directed graph structure as follows:

- First line includes two integers *M* and *K* that show the number of jobs and the number of direct precedence relations between them respectively
- Remaining next *M* lines are the related jobs by the fact that first job must be handled before the second and last line with "0 0" finishes the input

Here is the sample input txt file example:

```
5 4
1 2
2 3
1 3
1 5
0 0
```

And for illustration purpose, the respective Diagraph for above input file is:



As a hint, you can think this problem as an application of non-unique topological ordering in diagraphs. While ordering the jobs, you must check whether precedence constraints are violated or not. You may use the Digraph and related classes in our textbook and its web site (https://algs4.cs.princeton.edu/code/edu/princeton/cs/algs4/Digraph.java.html).

Your final output must print a possible order of getting all the jobs done with *M* integers as follows:

```
% java MyMain2 sampleinput2.txt
1 4 2 5 3
```

WHAT TO HAND IN

A zip file containing:

- ➔ The Java source of your programs.
- ➔ The Java sources should be WELL DOCUMENTED as comments, as part of your grade will be based on the level of your comments.

The zip file should be uploaded to Moodle.

IMPORTANT

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

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

2. 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.

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

4. For this assignment, you are allowed to use the codes given in our textbook and/or our lecture slides. You ARE NOT ALLOWED to use any codes from somewhere else (e.g., from the internet, other text books, other slides ...).

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

6. 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.

7. 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: MyDiaGraph

// Author: Ibrahim ILERI

// ID: 2100000000

// Section: 0

// Assignment: 3

// Description: This class defines a Diagraph

//-------------------------------------------------
```

8. Since your codes will be checked without your observation, you should report everything about your implementation. Well comment 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

  }
```

- Indentation, indentation, indentation...

9. This homework will be graded by your TAs, İbrahim İleri and Hamid Ahmadlouei (ibrahim.ileri@tedu.edu.tr, hamid.ahmadlouei@tedu.edu.tr). Thus, you may ask them your homework related questions through HW forum on Moodle course page. You are also welcome to ask your course instructor Tolga Çapın and Ulaş Güleç for help.

GRADING RUBRICs

PART I (60 points)						
Performance Element	**Weight**	**Master (4/4)**	**Advanced (3/4)**	**Developing (2/4)**	**Beginner (1/4)**	**Insufficient (0/4)**
Information	5%	Information (id, name, section, assignment number) given in each file.	Missing minor details.	Missing few details (e.g. section id).	Only name given.	None.
Documentation	5%	Every class and method has a detailed explanation (pre- and post-conditions, sample I/O, etc).	Most classes and methods have an explanation, but missing in some parts.	Attempted to document the classes and methods, but they are not clear.	Only few comments.	Not even an attempt.

Code Design	5%	Modular source code and code format. Complete submission of classes and methods.	Methods make sense. Includes constructor that initializes carefully.	Uses set/get methods as necessary.	Class does very little; most functions remain in one main (driver) class.	Methods not properly defined.
Abstract Data Type: Directed Graph	10%	Uses directed graph based data structure for implementation. And provides all functionality.	Uses directed graph based data structure for implementation. And provides most of expected functionalities.	Implements directed graph with major deviations from the specification.	Used different data structure from directed graphs to solve this problem.	Not even an attempt.
Functionality	65%	All functions are implemented with no missing functionality. Runs without any crash.	Missing some minor features or minor output problems. Runs without any crash.	Attempted to implement all functions but some of them do not work.	Only few functions are implemented correctly. Compiles but several warnings.	No working solution or does not compile.
Testing: Test data creation & generation	10%	Provided a tester class. Able to identify key test points. Clear on what aspects of the solution are being tested with each set. Able to generate test data automatically when necessary.	Provided a tester class. Able to identify key test points. Clear on what aspects of the solution are being tested with each set.	Provided a tester class. Able to identify key test points.	Sporadic.	No test case presented.

PART II (40 points)						
Performance Element	**Weight**	**Master (4/4)**	**Advanced (3/4)**	**Developing (2/4)**	**Beginner (1/4)**	**Insufficient (0/4)**
Information	5%	Information (id, name, section, assignment number) given in each file.	Missing minor details.	Missing few details (e.g. section id).	Only name given.	None.

Documentation	5%	Every class and method has a detailed explanation (pre- and post-conditions, sample I/O, etc).	Most classes and methods have an explanation, but missing in some parts.	Attempted to document the classes and methods, but they are not clear.	Only few comments.	Not even an attempt.
Code Design	5%	Modular source code and code format. Complete submission of classes and methods.	Methods make sense. Includes constructor that initializes carefully.	Uses set/get methods as necessary.	Class does very little; most functions remain in one main (driver) class.	Methods not properly defined.
Abstract Data Type: Directed Graph	10%	Uses directed graph based data structure for implementation. And provides all functionality.	Uses directed graph based data structure for implementation. And provides most of expected functionalities.	Implements directed graph with major deviations from the specification.	Used different data structure from directed graphs to solve this problem.	Not even an attempt.
Functionality	65%	All functions are implemented with no missing functionality. Runs without any crash.	Missing some minor features or minor output problems. Runs without any crash.	Attempted to implement all functions but some of them do not work.	Only few functions are implemented correctly. Compiles but several warnings.	No working solution or does not compile.
Testing: Test data creation & generation	10%	Provided a tester class. Able to identify key test points. Clear on what aspects of the solution are being tested with each set. Able to generate test data automatically when necessary.	Provided a tester class. Able to identify key test points. Clear on what aspects of the solution are being tested with each set.	Provided a tester class. Able to identify key test points.	Sporadic.	No test case presented.