CMPE 343

Fall 2021

Programming Homework II

This assignment is due by 23:55 on Wednesday 11 November 2020

Part I

Representing mazes as graphs.

In Part I of the assignment, you will write a Maze data type that can read **InputMaze** files, and print them out in the graph format. There are many different ways to represent a graph in Java. You are invited to write your own versions.

The maze InputMaze file Example

A maze input file is basically a text file that contains all the information you need to construct your maze. It consists of:

- **Number of rooms/vertices** An integer that tells you how many vertices are in the graph.
- **name x y** The next n lines each contain the name of a room (String) and its x and y coordinates (integers). The purpose of name in this example is strictly descriptive, you can use numbers instead in large examples.
- **room1 room2** After the list of vertices comes the edges/corridors. Each edge is listed as two integers, room1 and room2, the indices of its start and end vertices.
- **-1 -1** The .maze file will end with a line "-1 -1".

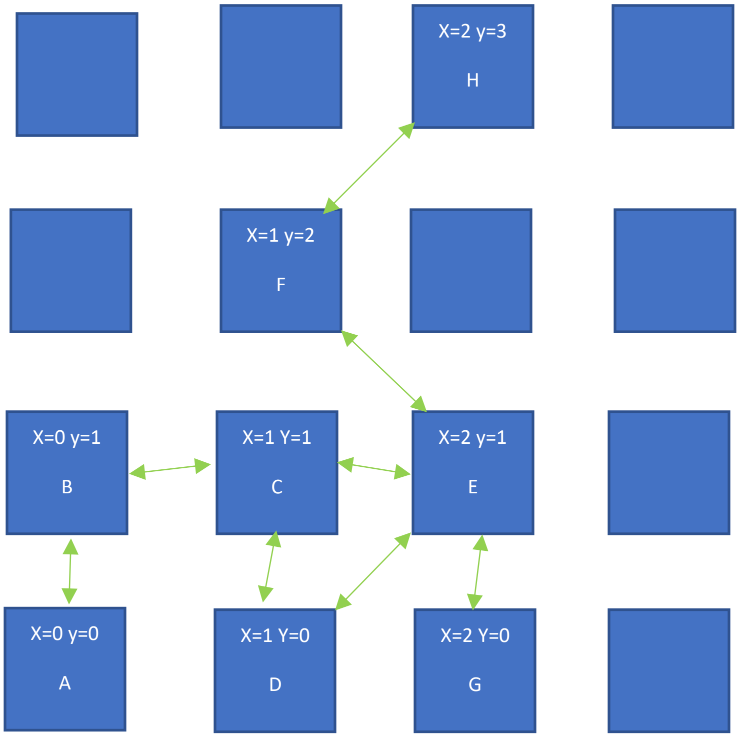
For example, here is a sample InputMaze file:

```
8
A 0 0
B 0 1
C 1 1
D 1 0
E 2 1
F 1 2
```

```
G 2 0

H 2 3


0 1

1 2

2 4

2 3

3 4

4 6

4 5

5 7

-1 -1
```

Example Maze Grid representation

green arrows show possible transition between rooms.

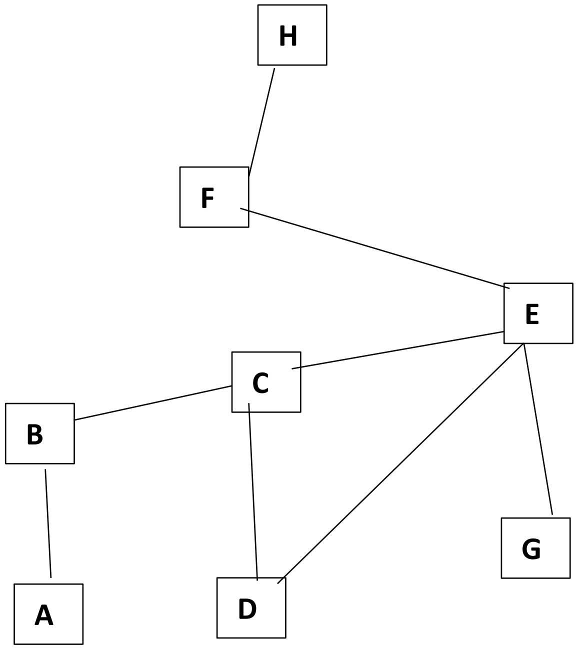


This is undirected graph. You can make the corridors in your maze two-way by adding edges in each direction.

You should write **PrintMaze** class to print out a maze as graph

```
% java PrintMaze input.maze
```

Example Maze Graph representation



Part II

The goal of Part II is to solve the maze your solution must build on your own code from Part I.

findPath()

Your goal is to find the first possible path from the first position in the maze to the last one by applying the search algorithm like sample output.

Example Maze Sample Output

```
A –> B –> C –> E –> F -> H
```

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,
- ➔ You will be asked to go through your program together with the TA or the instructor.

The zip file should be uploaded to Moodle.

IMPORTANT

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

1. This assignment is due by 23:55 on Wednesday 11 November 2020.
2. You should upload your homework to the course Moodle site before the deadline. No hardcopy submission is needed. You should include 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. Please see the course syllabus for further discussion of the late homework policy as well as academic integrity.
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. 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: Level tester class
// Author: Hamid Ahmadlouei
// ID: 2100000000
// Section: 1
// Assignment: 1
// Description: This class tests the …
//-----------------------------------------------------
```

6. 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...

- This homework will be graded by your TA, Hamid Ahmadlouei (hamid.ahmadlouei@tedu.edu.tr). Thus, you may ask him your homework related questions. You are also welcome to ask your course instructor for help.

GRADING RUBRICs

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	Methods make sense. Includes constructor	Uses set/get methods as necessary.	Class does very little; most functions remain in one main	Methods not properly defined.

		of classes and methods.	that initializes carefully.		(driver) class.	
PART I						
		Use graph based data structure implementation. And provides all functionality	Uses graph for implementation. And provides most of expected functionalities	Implements data structure with major deviations from the specification.	Used different data structure to solve this problem.	
	25%	All steps presented perfectly with details.	All steps presented with details but some are not clear.	Attempted to about all steps but half of them not clear with major deviations in details.	Attempted to few steps with no details	Not even an attempt.
PART II						
Functionality	55%	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 requirement but some of them do not work.	Only few steps 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 automaticall	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.

		y when necessary.				