Problem Solving Session

- The remainder of today's class will comprise the ***problem solving session*** (***PSS***).
- Your instructor will divide you into ***teams of 3 or 4 students***.
- Each team will ***work together*** to solve the following problems over the course of ***20-30 minutes***.
  - You may work on paper, a white board, or digitally as determined by your instructor.
  - You will submit your solution by pushing it to GitHub before the end of class.
- Your instructor will go over the solution before the end of class.
- If there is any time remaining, you will begin work on your homework assignment.

Class participation is a significant part of your grade (20%). This includes in class activities and the problem solving session.

Your Course Assistants will grade your participation by verifying that you pushed your solutions before the end of the class period each day.

Problem Solving Team Members

Record the name of each of your problem solving team members here.

Do not forget to ***add every team member's name***! Your instructor (or course assistant) may or may not use this to determine whether or not you participated in the problem solving session.

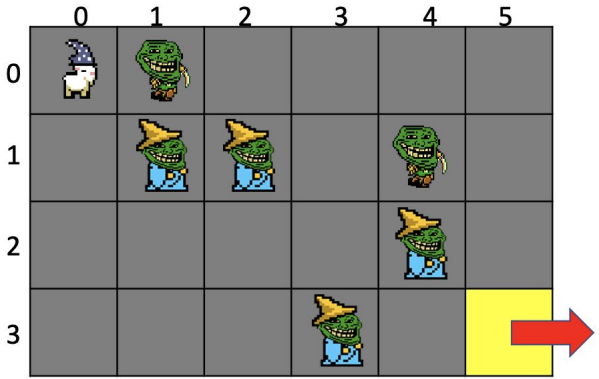
Dessa Shapiro
Mason Hendricks

Problem Solving 1

Problem Statement:

An infamous federal correctional facility is comprised of a grid of R x C jail cells, each devoid of natural light or fresh air. Lurking within some of these cells are ferocious trolls, the big beasts that wait for prisoners to come and then pounce and devour their prey. A prisoner named Mage is confined in the top leftmost cell of the correctional center. Each jail cell is connected to its neighbors by a door, providing the only means of movement throughout the facility. The only way out is through the bottom rightmost cell, known as the "escape room," which opens up to the outside world. Your task is to use DFS to find a safe path for Mage to reach the escape room.

Read through the problem statement above.
What is the string representation of a secure path that leads Mage to the escape room, using the example on the right?




```
[(0,0), (1,0) , (2,0), (2,1), (2, 2),
(2,3), (1,3), (0,3), (0,4), (0,5), (1,5),
(2,5), (3,5)]
```

```
public class Cell{
      Public Vertex<E>
      Public int x;
      Public int y;
      Public boolean troll;

      Public Cell(int x, int y, boolean troll) {
            this.x = x;
            this.y = y;
            This.hastroll = false;
      }


      public boolean isTroll() {
            return this.troll;
      }

      @Override
      public String toString()
            {return this.x +” ”+ this.y}
```

Problem Solving 2

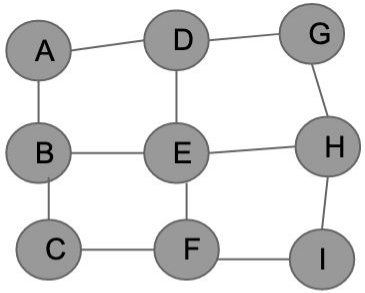
How would you represent a jail cell?
A cell has a location and may have a troll.
Write the Cell class at the left.

Problem Solving 3

On the right is the **visitDFPath** method in the AdjacencyGraph class.

Assuming that your solution visits neighbors in alphabetical order, what will **graph.visitDFPath(v, e, visited)** return, where **graph**, **v**, **e**, and **visited** are as follows?

graph =



v = A

e = I

visited = { A E F **}**

```
protected List<E> visitDFPath(Vertex<E> v, Vertex<E> e, Set<Vertex<E>> visited) {
    if (v == e) {
        List<E> path = new LinkedList<>();
        path.add(e.getValue());
        return path;
    } else {
        for (Vertex<E> neighbor : v.getNeighbors()) {
            if (!visited.contains(neighbor)) {
                visited.add(neighbor);
                List<E> path = visitDFPath(neighbor, e, visited);
                if (path != null) {
                    path.add(0, v.getValue());
                    return path;
                }
            }
        }
        return null;
    }
}
```

Returns{ A D G H I}

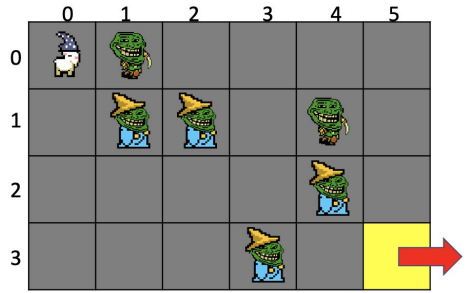
```
try (FileReader fr = new FileReader(filename);
    BufferedReader reader = new BufferedReader(fr)) {
    Int ROWS = integer.parseInt(reader.readline());
    Int COLS= integer.parseInt(reader.readline());
        Cell[][] cells = new Cell[ROWS][COLS];
String line = null;
Int row = 0;
this.trollCells = new HashSet<>();
while((line=reader.readLine())!=null) {
        String tokens = line.split(" ");
        for (int c= 0; c<col; c++) {
            char ch = tokens[col].charAt(0);
            if(ch == "T") {
                Cells[row][col] = new Cell(row, col, true)
                trollCells.add(cell[row][col]);
            } else {
            cells[row][col] = new Cell(row, col, false);
            }
            graph.add(cells[row][col]);
            If (col>0) {graph.connectUndirected(cells[row][col-1], cells[row][col])}
            If (row>0) {graph.connectUndirected(cells[row-1][col], cells[row][col])}
            }
            row++;
}
```

Problem Solving 4

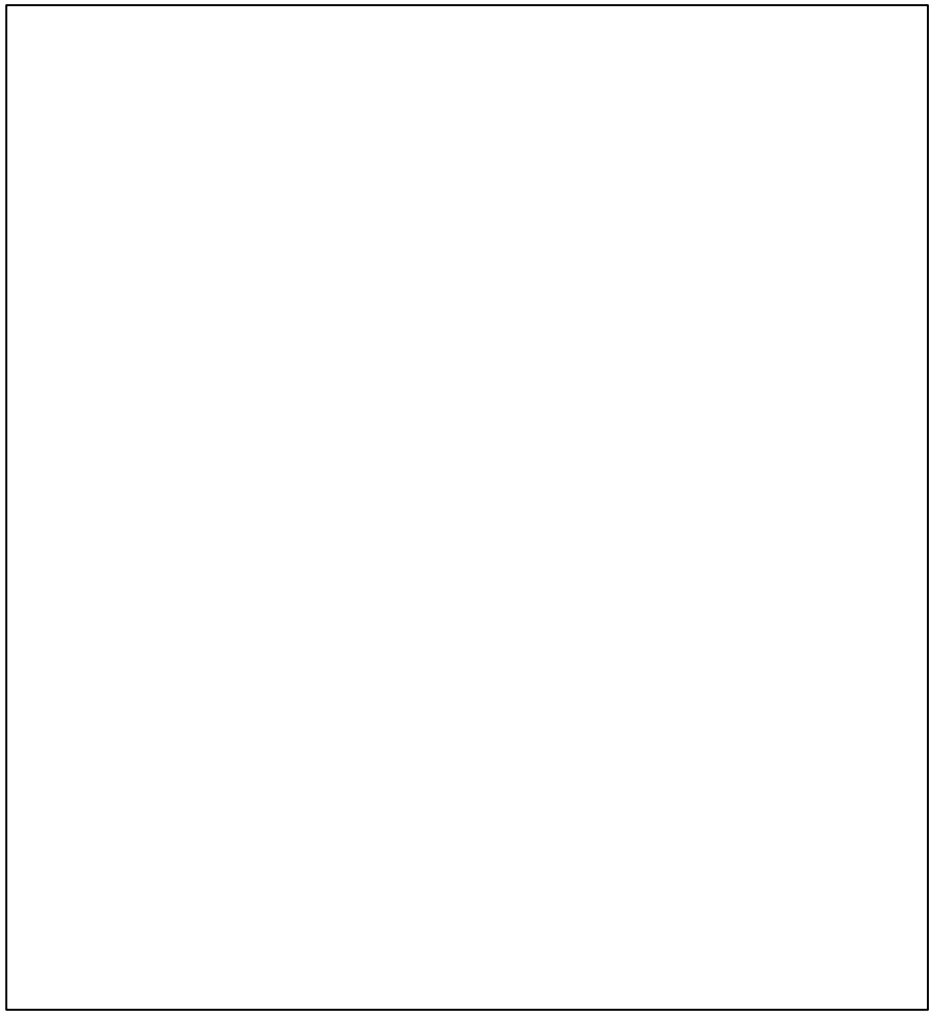
Write code on the left-hand side that parses a file containing a prison structure and then creates a graph representing it.

The format of a file for the accompanying prison structure is demonstrated in the following example.

```
4
6
E T E E E E
E T T E T E
E E E E T E
E E E T E E
```



The first two lines contain the number of rows and columns. Each of the remaining lines shows the content of cells in a row, where 'E' indicates the absence of a troll and 'T' indicates the presence of a troll.



Problem Solving 4 (cont)