CS 435 – Homework 7

Due for Section 001:	**10:00 AM on December 2, 2019**
Due for Section 003:	**11:30 AM on December 2, 2019**
Due for Section 005:	**10:00 AM on December 3, 2019**

About

This homework assignment asks you to apply single source shortest path (path finding) algorithms to a variety of problems.

There are 4 coding problems in this homework but **you don't need to do all of them!** You must complete problems 1 and 2 (Grid Walk and Heuristics), and you must complete **one** of either 3 or 4. You may choose which one.

Submissions

The written section of this assignment will be submitted in class on the due date. In the coding section, each problem will need to be submitted on HackerRank.

In order to submit work on HackerRank, you must join the homework 7 contest at https://www.hackerrank.com/cs435-f19-hw7. You may submit as many times as you would like, I will only look at the submission that you enter on Canvas.

In order to enter a submission on Canvas, first copy a permalink to your submission. The permalink is the URL of the page you are taken to after you click "Submit Code" on HackerRank - the page that shows Status as "Queued" or "Wrong Answer" or "Accepted" etc. Save that URL of your final submission for each challenge and submit it on Canvas for this homework.

Scoring

Your score on HackerRank will give you *an approximation* of what your grade for the coding section will be; however, I will read your code and determine your final grade myself, as detailed on the syllabus.

Problems

1	Grid Walk	20 points
2	Heuristics	20 points
3	Optimizing Routes	30 points
4	Essential Services	30 points
5	Written Problems	30 points
6	Extra Credit HW 7	1 points

1. Grid Walk (20 points)

https://www.hackerrank.com/contests/cs435-f19-hw7/challenges/grid-walk-1

Given a grid with obstacles, we want to find the path that gets you to the end in the shortest amount of time, using Dijkstra's algorithm. In this grid, you can move in all 8 directions: Up, Down, Left, Right, and the 4 diagonals.

You have been working out, and you can climb over obstacles, but it takes you longer. Moving to a grid location without an obstacle takes 1 minute. Moving to a grid location with an obstacle takes 2 minutes.

Input

- The first line contains an integer, n, the dimension of the nxn grid.
- The second line contains two space-separated integers, the starting position in the order row column.
- The third line contains two space-separated integers, the ending position in the order row column.
- The next n lines contain the space separated row. Each element is either a O indicating no obstacle or a X indicating an obstacle.

Constraints

You can assume a path exists (and therefore that the starting and ending positions are Os) and that the starting and ending positions are in-bounds.

Output

The output contains a single line with an integer, the shortest time it takes to reach the end.

Sample Input 1

```
4
0 2
3 1
X X O X
X X O O
X X X O
X O O O
```

Sample Output 1

```
4
```

Sample Input 2

```
6
0 1
5 2
O O O O O X
X X O X O O
O O X O O X
X O O O O O
O O X X X O
X O O X O O
```

Sample Output 2

```
5
```

2. Heuristics (20 points)

https://www.hackerrank.com/contests/cs435-f19-hw7/challenges/heuristics-1

Given a grid with obstacles, we want to find the shortest path from a starting position to an ending position. This time we'd like to use the A-Star algorithm. In this grid, you can only move in four directions: up, down, left, right therefore the heuristic that we will use will be Manhattan distance.

You've been hitting the gym, but have been taking it a bit easier recently. So you can still climb over obstacles, but it takes you even longer. Moving to a grid location without an obstacle takes 1 minute. Moving to a grid location with an obstacle takes 3 minutes.

Input

- The first line contains an integer, n, the dimension of the nxn grid.
- The second line contains two space-separated integers, the starting position in the order row column.
- The third line contains two space-separated integers, the ending position in the order row column.
- The next n lines contain the space separated row. Each element is either a O indicating no obstacle or a X indicating an obstacle.

Constraints

You can assume a path exists (and therefore that the starting and ending positions are Os) and that the starting and ending positions are in-bounds.

Output

The output contains a single line with an integer, the length of the shortest path.

Sample Input 1

```
4
0 2
3 1
X X O X
X X O O
X X X O
X O O O
```

Sample Output 1

```
6
```

Sample Input 2

```
6
0 1
5 2
O O O O O X
X X O X O O
O O X O O X
X O O O O O
O O X X X O
X O O X O O
```

Sample Output 2

```
8
```

3. Optimizing Routes (30 points)

https://www.hackerrank.com/contests/cs435-f19-hw7/challenges/optimizing-routes-1

I have my set ways of traveling to places I visit often. However, I'm curious if my routes are actually the best routes according to maps. Determine the difference in my route and the supposed best route for each of my destinations.

Starter code

The code below will help you read input for this problem.

- C++: https://repl.it/@ncantor/Optimizing-Routes-cpp
- Java: https://repl.it/@ncantor/Optimizing-Routes-java
- Python: https://repl.it/@ncantor/Optimizing-Routes-py

Input

- The first line will contain a location, my home.
- The next line will contain an integer n, the number of places I visit often enough to have a set route, followed by how long my route to that place takes.
- The next line will contain an integer, r, the number of roads.
- Finally, the last r lines will contain the roads, consisting of comma-separated endpoints followed by the time it takes to travel the road.

Constraints

You can assume all the weights are positive (it takes time to travel a road) and that all locations have unique names. Roads in this scenario are all one-way, so each road is a directed edge {from, to}. You may choose which appropriate path-finding algorithm to use to solve this problem.

Output

- The output will consist of n lines.
- Each line consists of a destination I visit often, followed by a space, followed by either "FASTEST" if my route is equal to or faster than the recommended route. "NO PATH" if there is no recommended route from my home to that location. Or an integer x representing how much faster the recommended route is.
- The n destinations will be ordered alphabetically, case-insensitive. This means NJIT will come before Northampton.

See the sample output section and hackerrank for concrete examples

Sample Input 1

```
Home
4
Library, 9
NJIT, 16
NYC, 6
Atlantic_City, 21
17
Museum, Library, 3
Museum, Newark_Downtown, 3
Museum, Halsey_St, 2
Museum, Home, 5
Home, Trenton, 17
Home, Ironbound, 12
Ironbound, NJIT, 3
Home, NJIT, 16
Home, NYC, 6
Home, Subway, 10
Home, Halsey_St, 4
Home, Kean, 10
NJIT, Kean, 7
Atlantic_City, Subway, 12
NYC, Halsey_St, 5
Halsey_St, Home, 4
Trenton, Atlantic_City, 4
```

Sample Output 1

```
Atlantic_City FASTEST
Library NO PATH
NJIT 1
NYC FASTEST
```

4. Essential Services (30 points)

https://www.hackerrank.com/contests/cs435-f19-hw7/challenges/all-pairs-1

King's Landing is trying to evaluate the quality of life of its citizens. One of the components is the time it takes each citizen to get to every essential service. Given the citizens and services, find the distance from each citizen to each service.

Starter code

The code below will help you read input for this problem.

- C++: https://repl.it/@ncantor/Essential-Services-cpp
- Java: https://repl.it/@ncantor/Essential-Services-java
- Python: https://repl.it/@ncantor/Essential-Services-py

Input

- The first line contains an integer, c, the number of citizens.
- The next c lines contain the citizens' names.
- The next line contains an integer, s, the number of services.
- The next s lines contain the services' names.
- The next line contains an integer, e, the number of direct routes.
- Each of the next e lines contains a direct route which consists of two comma-separated entities (citizens or services) and a time it takes to get from one to the other.

Constraints

You can assume all the weights are positive (it takes time to get from one place to the other) and that all citizens and services have unique names.

Output

- The output will consist of $c + 1$ lines.
- The first line consists of the services in alphabetical order, space-separated.
- The next c lines consist of a citizen and their times from each service. You should have s space-separated integers, the time from each service in alphabetical order, followed by the citizen name (in alphabetical order).

See the sample output section and hackerrank for concrete examples

Sample Input 1

```
5
Cersei
Robert
Jamie
Qyburn
Tomin
2
Castle
Church
9
Castle , Robert , 15
Robert , Qyburn , 155
Robert , Tomin , 15
Castle , Tomin , 35
Qyburn , Church , 15
Robert , Jamie , 30
Jamie , Church , 115
Robert , Cersei , 300
Cersei , Church , 60
```

Sample Output 1

```
Castle Church
220 60 Cersei
45 115 Jamie
170 15 Qyburn
15 145 Robert
30 160 Tomin
```

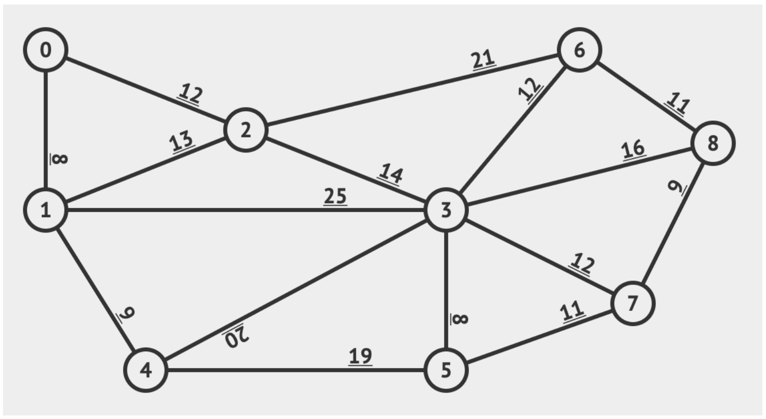
Name: ______________________________

Section: ______________________________

5. Written Problems (30 points)

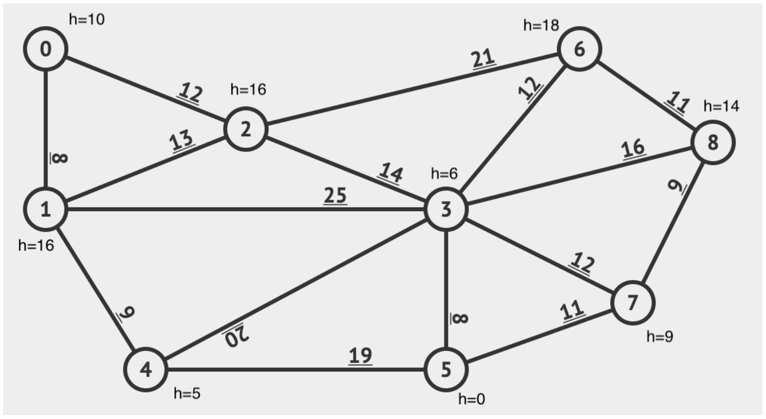
1. (5 points) What makes Dijkstra's a Greedy algorithm? Justify your answer using both properties of Greedy algorithms.

2. (10 points) Run Dijkstra's on the below graph from starting node 1 to destination node 5. Show your work at each iteration in a distances table.



3. (5 points) How does A* differ from Dijkstra's?

4. (10 points) Run A* on the below graph from starting node 1 to destination node 5. Use the heuristics given. Start at node 1, Show your work at each iteration in a distances table.



6. Extra Credit HW 7 (1 points)

Solve the challenge here: https://www.hackerrank.com/homework-7-extra-credit

Note that the main difficulty in this problem is probably going to be efficiency. An inefficient solution will likely hit timeouts.