**CS1083**

**Assignment #8**

**Daniyal Khan**

**3765942**

**EmerTraversal.java:**

import *java*.*io*.*\**;

import *java*.*util*.*Scanner*;

*public* *class* EmerTraversal {

*static* Scanner scan = *new* *Scanner*(System.*in*);

*public* *static* void *main* (String[] args) {

char[][] map = *map*(args); // *get the map from the csv file*

*rainfallOccurs*(map); // *change the map according to the flood*

*traversal*(map); // *traversing the map to get to see if emergency vechile at a location can get to a specific spot*

}

*public* *static* void *traversal*(char[][] map) {

System.*out*.*println*("Row Coordinate of Vehicles: ");

int vehicleRow = Integer.*parseInt*(scan.*nextLine*());

System.*out*.*println*("Column Coordinate of Vehicles: ");

int vehicleColumn = Integer.*parseInt*(scan.*nextLine*());

System.*out*.*println*("Row Coordinate of Emergency: ");

int emergencyRow = Integer.*parseInt*(scan.*nextLine*());

System.*out*.*println*("Column Coordinate of Emergency: ");

int emergencyColumn = Integer.*parseInt*(scan.*nextLine*());

boolean[][] search = *new* boolean[map.*length*][map[0].*length*];

System.*out*.*println*();

*if* (*pathExists*(map, search, vehicleRow, vehicleColumn, emergencyRow, emergencyColumn)) {

System.*out*.*println*("There does exist a land path to the emergency");

} *else* {

System.*out*.*println*("No land path to the emergency exists");

}

}

*public* *static* boolean *pathExists*(char[][] map, boolean[][] search, int currentRow, int currentColumn, int destinationRow, int destinationColumn) {

*if* (currentRow == destinationRow && currentColumn == destinationColumn) {

search[currentRow][currentColumn] = true;

*return* true;

} *else* {

boolean path = false;

// *Down*

*if* (currentRow + 1 < map.*length* && (map[currentRow+1][currentColumn] != 'R' && map[currentRow+1][currentColumn] != 'L') && !search[currentRow+1][currentColumn]) {

search[currentRow+1][currentColumn] = true;

path = *pathExists*(map, search, currentRow+1, currentColumn, destinationRow, destinationColumn);

}

// *Up*

*if* (currentRow - 1 >= 0 && (map[currentRow-1][currentColumn] != 'R' && map[currentRow-1][currentColumn] != 'L') && !search[currentRow-1][currentColumn]) {

search[currentRow-1][currentColumn] = true;

path = path || *pathExists*(map, search, currentRow-1, currentColumn, destinationRow, destinationColumn);

}

// *Right*

*if* (currentColumn + 1 < map[0].*length* && (map[currentRow][currentColumn+1] != 'R' && map[currentRow][currentColumn+1] != 'L') && !search[currentRow][currentColumn+1]) {

search[currentRow][currentColumn+1] = true;

path = path || *pathExists*(map, search, currentRow, currentColumn+1, destinationRow, destinationColumn);

}

// *Left*

*if* (currentColumn - 1 >= 0 && (map[currentRow][currentColumn-1] != 'R' && map[currentRow][currentColumn-1] != 'L') && !search[currentRow][currentColumn-1]) {

search[currentRow][currentColumn-1] = true;

path = path || *pathExists*(map, search, currentRow, currentColumn-1, destinationRow, destinationColumn);

}

*return* path;

}

}

*public* *static* void *rainfallOccurs*(char map[][]) {

System.*out*.*println*("Enter rainfall severity level: ");

int value = Integer.*parseInt*(scan.*nextLine*());

*for* (int i = 0; i < map.*length*; i++) {

*for* (int j = 0; j < map[0].*length*; j++) {

*if* (map[i][j] == 'R') {

*flood*(map, value, i, j, 'R');

}

}

}

*for* (int i = 0; i < map.*length*; i++) {

*for* (int j = 0; j < map[0].*length*; j++) {

*if* (map[i][j] == 'L') {

*flood*(map, value, i, j, 'L');

}

}

}

System.*out*.*println*();

*printArray*(map);

System.*out*.*println*();

}

*public* *static* void *flood*(char[][] map, int severityLevel, int row, int column, char waterBody) {

*if* (waterBody == 'R') { // *if waterbody is a river*

// *Down*

*if* (row + 1 < map.*length* && Character.*getNumericValue*(map[row+1][column]) <= severityLevel) {

map[row+1][column] = waterBody;

*flood*(map, severityLevel, row+1, column, waterBody);

}

// *Up*

*if* (row - 1 >= 0 && Character.*getNumericValue*(map[row-1][column]) <= severityLevel) {

map[row-1][column] = waterBody;

*flood*(map, severityLevel, row-1, column, waterBody);

}

// *Right*

*if* (column + 1 < map[0].*length* && Character.*getNumericValue*(map[row][column+1]) <= severityLevel) {

map[row][column+1] = waterBody;

*flood*(map, severityLevel, row, column+1, waterBody);

}

// *Left*

*if* (column - 1 >= 0 && Character.*getNumericValue*(map[row][column-1]) <= severityLevel) {

map[row][column-1] = waterBody;

*flood*(map, severityLevel, row, column-1, waterBody);

}

} *else* *if* (waterBody == 'L') { // *if waterbody is a lake*

int floodTiles = severityLevel; // *Lakes will only flood adjacent altitude areas in increments of 2*

*if* (severityLevel%2 != 0) { // *so if Severity Level is odd then flooded tiles would be one altitude below it*

floodTiles = --severityLevel;

}

// *Down*

*if* (row + 1 < map.*length* && Character.*getNumericValue*(map[row+1][column]) <= floodTiles) {

map[row+1][column] = waterBody;

*flood*(map, severityLevel, row+1, column, waterBody);

}

// *Up*

*if* (row - 1 >= 0 && Character.*getNumericValue*(map[row-1][column]) <= floodTiles) {

map[row-1][column] = waterBody;

*flood*(map, severityLevel, row-1, column, waterBody);

}

// *Right*

*if* (column + 1 < map[0].*length* && Character.*getNumericValue*(map[row][column+1]) <= floodTiles) {

map[row][column+1] = waterBody;

*flood*(map, severityLevel, row, column+1, waterBody);

}

// *Left*

*if* (column - 1 >= 0 && Character.*getNumericValue*(map[row][column-1]) <= floodTiles) {

map[row][column-1] = waterBody;

*flood*(map, severityLevel, row, column-1, waterBody);

}

}

}

*public* *static* char[][] *map*(String[] args) {

*try* {

File fileIn = *new* *File*(args[0]);

Scanner scan = *new* *Scanner*(fileIn);

int rows = Integer.*parseInt*(scan.*nextLine*());

int columns = Integer.*parseInt*(scan.*nextLine*());

char[][] map = *new* char[rows][columns];

*for* (int i = 0; i < rows; i++) {

String line = scan.*nextLine*(); // *Reading each line*

String[] values = line.*split*(","); // *Splitting the line by commas*

*for* (int j = 0; j < columns; j++) {

map[i][j] = values[j].*charAt*(0);

}

}

// *printArray(map);*

*return* map;

} *catch* (FileNotFoundException fnfe) {

System.*out*.*println*(fnfe.*getMessage*());

} *catch* (IndexOutOfBoundsException iofbe) {

System.*out*.*println*("You must pass in a file name with the run command!");

}

*return* null;

}

*public* *static* void *printArray*(char[][] array) {

*for* (int i = 0; i < array.*length*; i++) {

*for* (int j = 0; j < array[0].*length*; j++) {

System.*out*.*print*(array[i][j] + " ");

}

System.*out*.*println*();

}

}

}

**Output:  
  
A computer screen shot of a computer

Description automatically generated**

**A computer screen shot of a computer

Description automatically generated**

**A computer screen shot of a computer

Description automatically generatedA computer screen with white text

Description automatically generated**