CSCI251 Assignment 1 Program Document

# Program Requirements:

A brief explanation for this assignment is as such, this program will firstly read a config file to obtain the grid indices followed by 3 file names. The 3 files will contain respectively the city location, cloudiness index values and pressure index values. The 3 file names will then be used to open the listed files and obtain the grid’s X and Y coordinates and the value stored within. The values will then be stored in a 3D array and the implementation will be further elaborated later. The final requirement is to calculate the rain probability of the cites by cross referencing the cloud and pressure values on the respective indices.

The first main function is to prompt a user to input a file name where the config details can be found. All grid values found are preceded with a GridX\_IdxRange or GridY\_IdxRange for consistency, it will then be further assumed that the next 3 lines of text in the config file ending with “.txt” will be the city location values, cloud cover values and pressure values respectively. My program will then take in the file names and proceed to open and store the values respective to their X, Y values into a dynamic 3D array. The array has to be dynamic as the grid indices can only be known during runtime.

The second function will be to display the values stored in a transposed grid formation representing a map which essentially is a 2D array. The map has to be transposed as most conceptualizations of 2D arrays have the 0,0 coordinates at the top left of the diagram whilst this program displays all its values with the 0,0 coordinates at the bottom left of the diagram. The values represented in the map for cloud and pressure is an int casted to the value divided by 10, as the grid only represents the value in the ten’s position. There is also a sub function to display the values in (L,M,H) symbols. As for city values, the City ID is only displayed while the areas not occupied by any city will be blank.

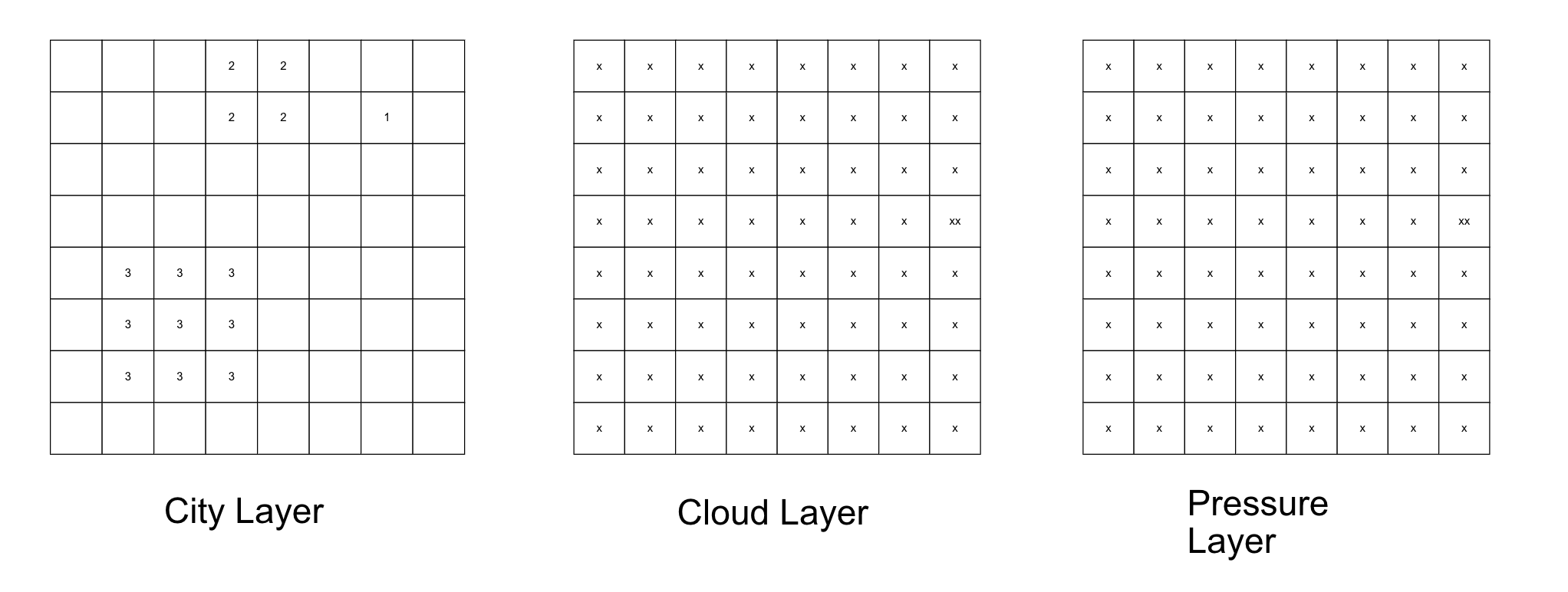
The final function will be to compute the rain probability and print a rain summary report.

This can be achieved by cross referencing the cloud and pressure values on the city’s location whist +1 of its surroundings and extrapolating the rain probability with the table given in the assignment pdf to derive an accurate probability of each city in the grid.

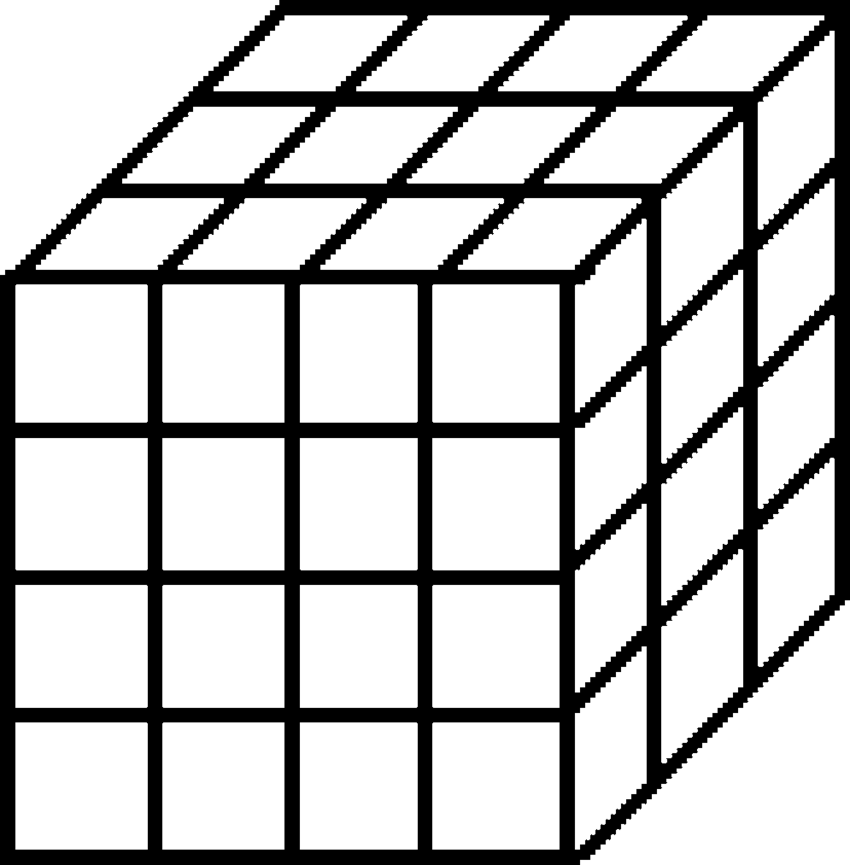
# Program Design:

To store all the different layers of mapped values for this program, I have decided to use a 3D dynamic array and pointer arithmetic to traverse the array.

2D blow out view of the 3 “layers”:



3D conceptualization of the array model:



Pressure layer

Cloud layer

City layer

Therefore, as shown above for cross referencing data on the same grid, a simple “layer change” would be sufficient to access the different value types on the same coordinates.

This method of storing the 3 layers of mapped values on a 2D grid/map is essentially the crux of the whole program. This also allows of efficiency for certain cases where during traversal and a different “layer” value is needed for example, I am currently traversing the array on the city level and require the cloud layer value of same location as the CityID, I would simply change the first index layer ‘x’ (3Darray[x][y][z]) to access the cloud value of the respective coordinates.

# Summary on each program module:

## Read in and process a config file

This module reads in a config file that matches the name of the user’s input. For testing the file that was used was “config.txt”. The file should consist of 2 Grid values of X and Y respectively. The format of the value should follow “GridX\_IdxRange=0-8” or “GridY\_IdxRange=0-8” where the value preceded by the “– “will be used for the X or Y index respectively. Upon extracting the X and Y indices, the module will then create a 3D dynamic array as such 3DArray[3][X][Y], the first index is always 3 due to the fact it’s an aggregate of 3 2D maps. Additionally, all values in the new 3D array will be normalized to 0 to avoid garbage values being displayed during the other modules.





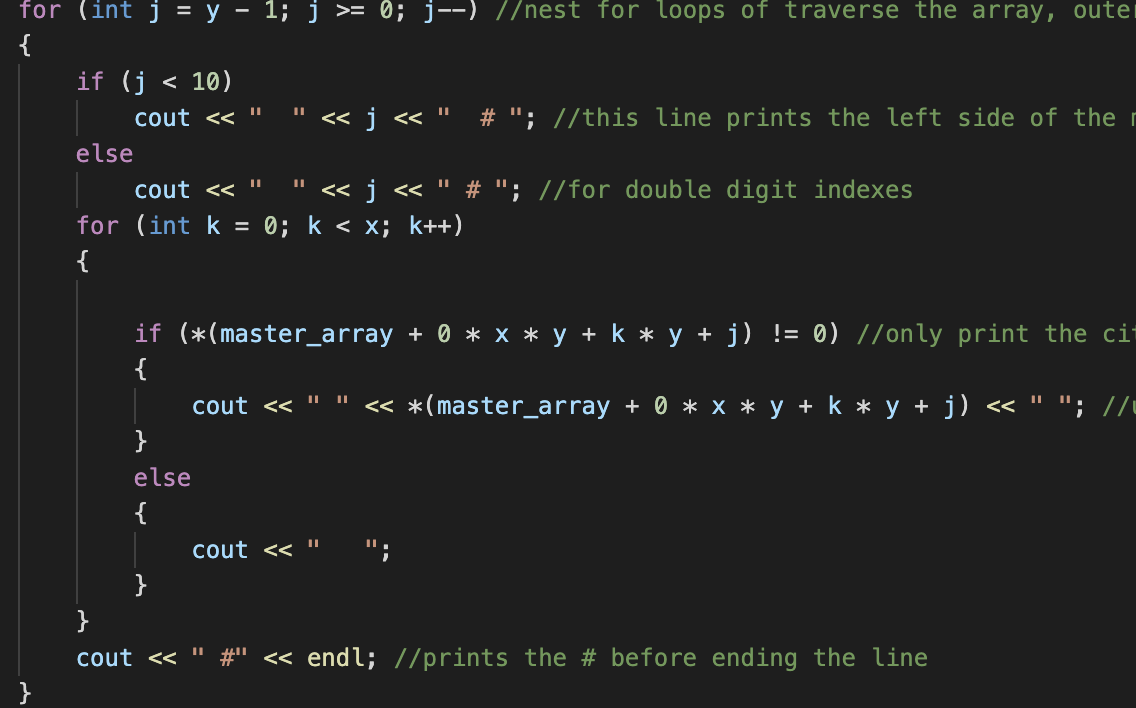
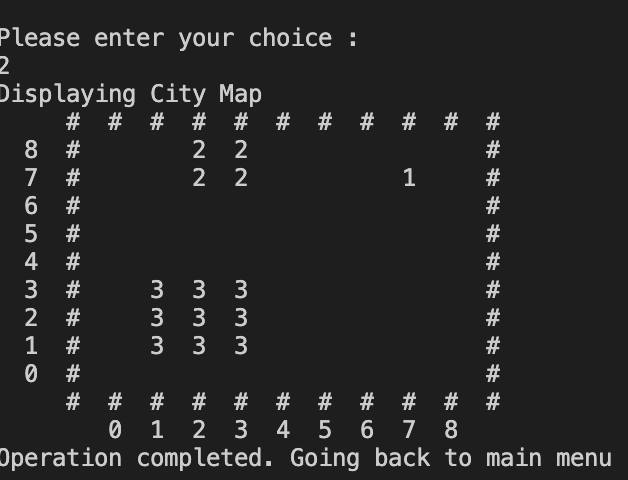
The module will then further read line by line in the config file and search for a text match of a line of string that ends with a “.txt”. what is preceded by the “.txt” is omitted as the test cases that will be provided by the lecturer will be this specific order [City, Cloud, Pressure] regardless of file name so long as it is a text file. Upon hitting a “.txt” match, the module will further call another read file function and read line by line till end of file. At every line of .txt files will have an X, Y and Value in this format ([1, 1]-3-Big\_City) for city file or ([1, 1]-80) for cloud and pressure file. As the format has the x and y coordinated, it will directly map the values into the 2D array of the respective data type (city, cloud, pressure).

## Display city map

This module displays the location of each city on the map by traversing the array and displaying the output in the terminal if there is a cityID value present or a blank space if there is none, the method of traversal is using pointer arithmetic and nested for loops.

Additionally, to print the top and the bottom of the map, a for loop is needed to be able to match the contents of the map as the X and Y indices are only known during run time.

The entire array will also need to be transposed as traditionally most 2D array outputs have the 0,0 coordinates at the top left of the display. But for this program, the 0,0 coordinates are the the bottom of the display. Therefore, the map needs to be transposed by having the Y index decremented to 0 in the outer loop and X index incremented to X from 0 in the inner loop as such.

## Display cloud coverage map(index/LMH)

## Display pressure coverage map(index/LMH)

## Show Weather Forecast summary report

## Rain probability map(index/LMH)

## Rain probability of the entire map

# Reflections