Processing Configuration File Information

Upon startup, program should prompt user for the following

Please enter config filename : config.txt
When user presses 'Enter', the program should perform File I/O to read in the contents in the

stated filename (e.g. in this case 'config.txt')

The config.txt contains data like size of grid-map area (indicated by index-ranges), as well as a series of filenames, which your program should further access, in order to read in the various relevant weather data provided by the Meteorological Station. Figure A-1 below provides a sample of the actual contents within this configuration file.

```
// The range of 'horizontal' indices, inclusive
// E.g. if the range is 0-4, then the indices are 0, 1, 2, 3, 4
GridX_IdxRange=0-8

// The range of 'vertical' indices, inclusive
// E.g. if the range is 0-3, then the indices are 0, 1, 2, 3
GridY_IdxRange=0-8

// [x,y] grid-areas which are occupied by cities
citylocation.txt

// "next day" forecasted cloud coverage (%) for
// each [x,y] grid-area
cloudcover.txt

// "next day" forecasted atmospheric pressure intensity (%) for
// each [x,y] grid-area
pressure.txt
```

Fig. A-1

Note : The ranges for both '`GridX_IdxRange`' and '`GridY_IdxRange` ' will determine the size of the display map. The ranges' values are editable!

Map-Grid Coordinate System (used by Meteorological Station) City Map Input Data & Output Requirements

Figure B-1 below provides a sample of the **actual contents** within the input file (e.g. `citylocation.txt`), storing city-occupied grid areas.

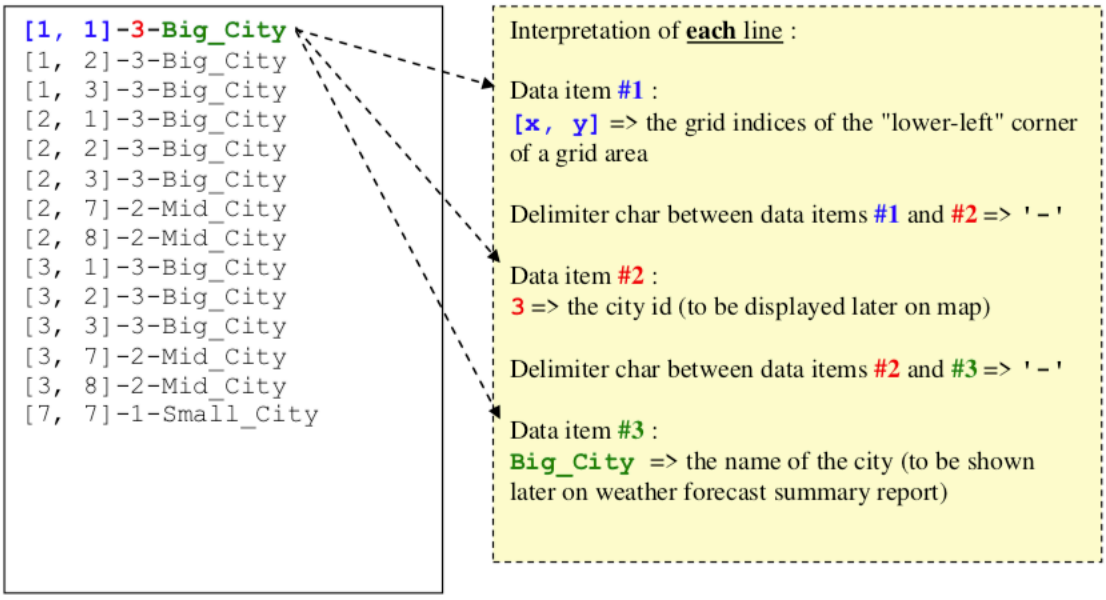


Fig. B-1

To aid in the visualization and understanding of how the city location input data will look like, in a 2D graphical format, please refer to the figure B-2 below.

Each [x, y] read in from the input file (see earlier **Fig. B-1**) describes the indices of the LOWER LEFT corner of a grid area

E.g. [3, 8] refers to the Lower Left corner of this grid area

[0, 0] origin refers to the Lower Left corner of this grid area

The range of (horizontal) indices to be shown on the map ranges from 0-8 (inclusive)

This info is retrieved from the value stored in the configuration file (see **Appendix A**, **Fig. A-1**), assigned to the variable `'GridX_IdxRange'`

The same treatment applies for the variable `'GridY_IdxRange'`, for the range of vertical indices

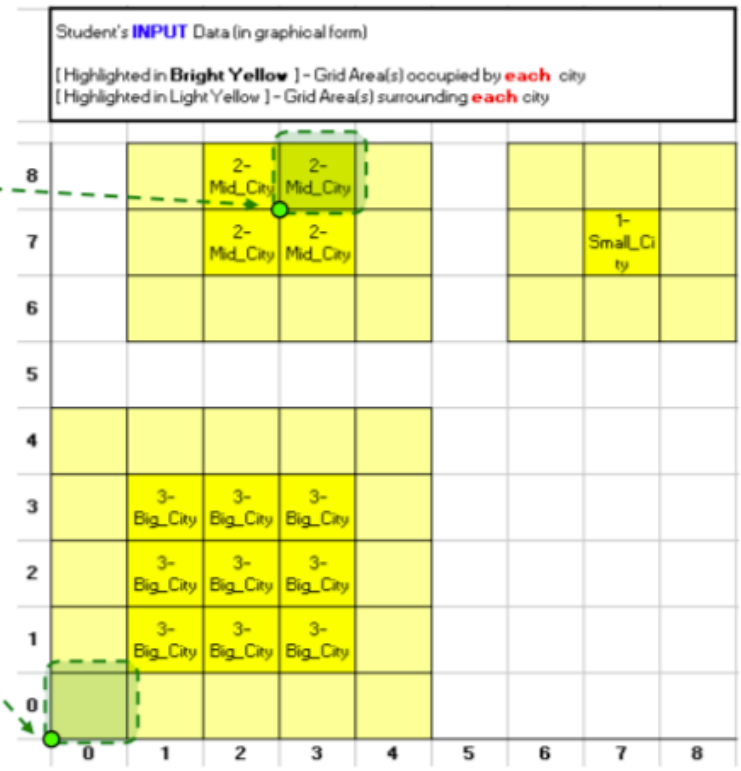


Fig. B-2

Note :

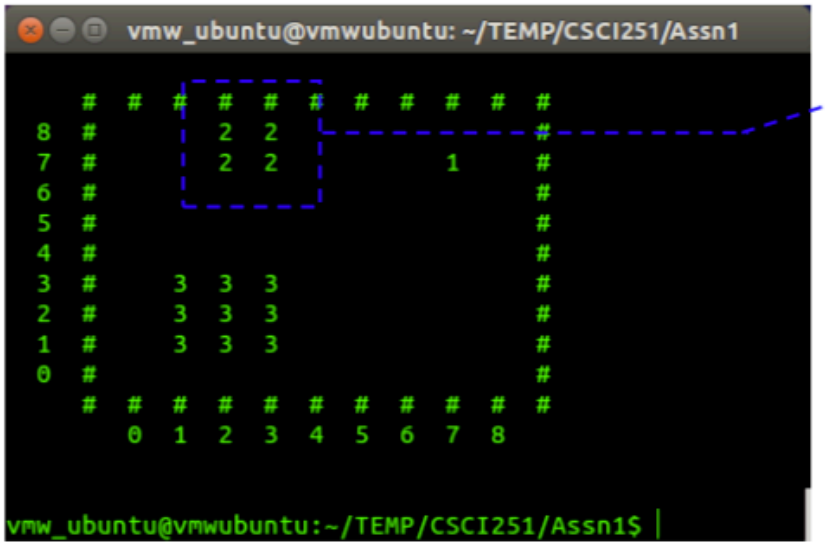
a) Bright Yellow grid areas indicate the ACTUAL Grid Area occupied by a particular city. For example, in the **fig. B-2** above, the city named "Mid_City" occupies the following grid areas : [2, 7], [2, 8], [3, 7], [3, 8]

b) Light Yellow grid areas indicate the grid areas surrounding the perimeter of EACH city. For example, in **fig. B-2** above, the city named "Mid_City" has the following grid areas surrounding its perimeter : [1, 6], [1, 7], [1, 8], [2, 6], [3, 6], [4, 6], [4, 7], [4, 8]

- The grid areas [1, 9], [2, 9], [3, 9], [4, 9] are not shown as they are **BEYOND the upper limits** of the vertical `GridY_IdxRange` !

Processing requirements - Display City Map

Realistically, there is limited graphical display capabilities available when you are constrained to displaying output on Ubuntu shell's terminal. Figure B-3 below illustrates the actual display formatting requirements when user selects the "Display City Map" option from your program's menu.



Note :

(i)
The **city id** is displayed on the relevant grid areas, instead of city names!

(ii)
Refer to **Fig. B-1**, which describes how each city's id value is stored in the input file

(iii)
The entire extent of the map is bounded by the '#' char !

Fig. B-3

a) As discussed in earlier **Appendix A**, **Fig. A-1**, the range values for variables '`GridY_IdxRange`' and '`GridY_IdxRange`' can vary.

This implies the **size** and **shape** of your map display can change as well (e.g. it is possible to have a large 'rectangular' shaped grid-area map!).

Therefore, please do not assume / "hard code" any constant values for your array sizes to store your map data!

b) You **must** make use of basic C++ array, and dynamic memory allocation (research on how to use '`new`') to initialize the size of your arrays during program runtime, to store all relevant weather map data.

c) The type of the array (e.g. `int`, `double`, `struct`, etc) is up to you, and there is no restriction on how many arrays you think is necessary to store all the relevant weather data.

d) Before your program exit, you **must** deallocate all memory that was dynamically allocated by you during runtime (research on how to use '`delete`'). Failure to do so results in memory leak ... and marks deduction!

Cloud Cover Input Data & Output Requirements

Figure C-1 below provides a sample of the **actual contents** within the input file (e.g. `cloudcover.txt`), storing the cloud cover input data, for each grid area in the map.

Note : due to lack of "vertical space" on this page, the entire content is split across Figures **C-1a**, **C-1b** and **C-1c**. In reality, the data in all 3 figures are combined and stored in a single input file!

Fig. C-1a

```
[0, 0]-41
[0, 1]-93
[0, 2]-90
[0, 3]-24
[0, 4]-70
[0, 5]-39
[0, 6]-47
[0, 7]-35
[0, 8]-83
[1, 0]-38
[1, 1]-66
[1, 2]-45
[1, 3]-11
[1, 4]-53
[1, 5]-35
[1, 6]-88
[1, 7]-75
[1, 8]-21
[2, 0]-56
[2, 1]-81
[2, 2]-34
[2, 3]-76
[2, 4]-53
[2, 5]-44
[2, 6]-70
[2, 7]-38
[2, 8]-32
[3, 0]-86
[3, 1]-13
[3, 2]-23
[3, 3]-93
[3, 4]-68
[3, 5]-26
[3, 6]-53
[3, 7]-52
[3, 8]-29
```

Fig. C-1b

```
[4, 0]-76
[4, 1]-60
[4, 2]-43
[4, 3]-82
[4, 4]-40
[4, 5]-72
[4, 6]-48
[4, 7]-29
[4, 8]-75
[5, 0]-16
[5, 1]-49
[5, 2]-36
[5, 3]-53
[5, 4]-18
[5, 5]-47
[5, 6]-27
[5, 7]-98
[5, 8]-78
[6, 0]-68
[6, 1]-63
[6, 2]-33
[6, 3]-92
[6, 4]-27
[6, 5]-48
[6, 6]-13
[6, 7]-15
[6, 8]-37
[7, 0]-47
[7, 1]-3
[7, 2]-8
[7, 3]-17
[7, 4]-62
[7, 5]-62
[7, 6]-14
[7, 7]-35
[7, 8]-84
```

Interpretation of **each** line :

Data item **#1** :
`[x, y]` => the grid indices of the "lower-left" corner of a grid area

Delimiter char between data items **#1** and **#2** => `'-'`

Data item **#2** :
`76` => the "next day" forecast of cloud cover for the grid area.

Max value : **99**
- means the grid area is forecasted to be entirely covered with thick clouds!

Min value : **0**
- means not a single wisp of cloud - "clear blue skies" for the grid area

Fig. C-1c

```
[8, 0]-7
[8, 1]-23
[8, 2]-63
[8, 3]-24
[8, 4]-37
[8, 5]-18
[8, 6]-44
[8, 7]-6
[8, 8]-18
```

To aid in the visualization and understanding of how the cloud cover input data will look like, in a 2D graphical format, please refer to the figure C-2 below.

Referring to the FIRST LINE shown in earlier **Fig. C-1a**, the grid area [0, 0] has a cloud cover value of 41, which is shown here.

Referring to the FIRST LINE shown in earlier **Fig. C-1b**, the grid area [0, 4] has a cloud cover value of 76, which is shown here.

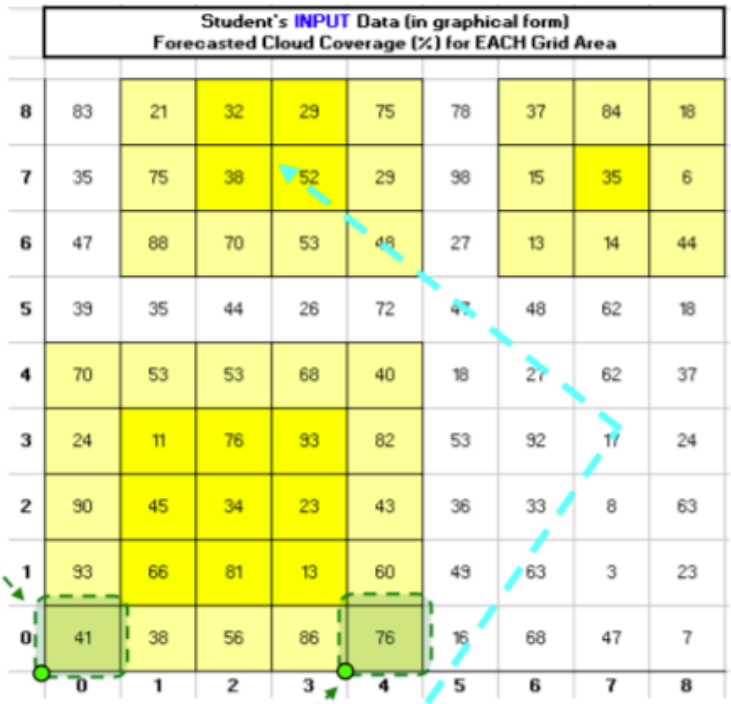


Fig. C-2

Note :

a) Bright Yellow grid areas indicate the ACTUAL Grid Area occupied by a particular city. For example, in the **fig. C-2** above, the city named "Mid_City" occupies the following grid areas : [2, 7], [2, 8], [3, 7], [3, 8]

b) Light Yellow grid areas indicate the grid areas surrounding the perimeter of EACH city. For example, in **fig. C-2** above, the city named "Mid_City" has the following grid areas surrounding its perimeter : [1, 6], [1, 7], [1, 8], [2, 6], [3, 6], [4, 6], [4, 7], [4, 8]

- The grid areas [1, 9], [2, 9], [3, 9], [4, 9] are not shown as they are **BEYOND the upper limits** of the vertical `GridY_IdxRange` ! As a result, they are ***not included*** in the **ACC** computation (see below) as well ...

c) For each city, the AVERAGE CLOUD COVER (ACC) value is derived, by the following :

ACC = SUM (cloud cover values of city + surrounding grid areas) / Total # of grid areas

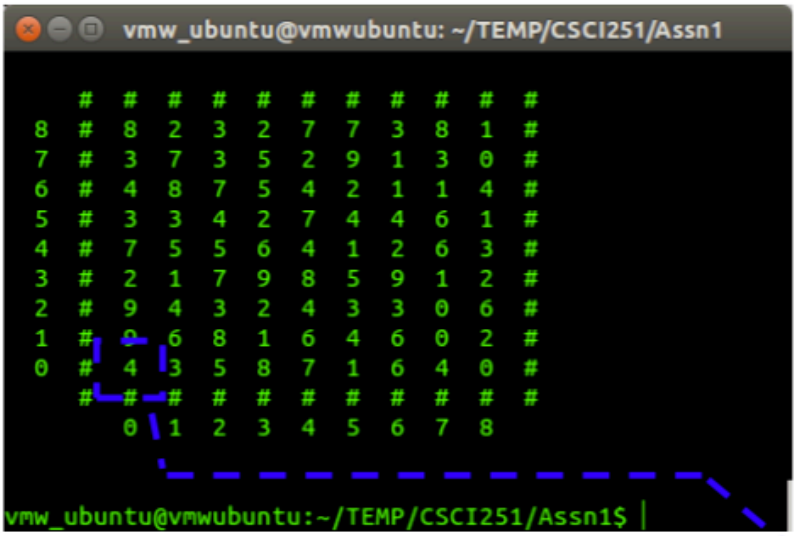
For example, for the city named "Mid_City" :

ACC = ((38 + 32 + 52 + 29) + (88 + 75 + 21 + 70 + 53 + 48 + 29 + 75)) / 12

= **50.83**

Processing requirements - display cloud coverage map (cloudiness index)

Figure C-3 below illustrates the actual display formatting requirements when user selects the "Display Cloud Coverage Map (cloudiness index)" option from your program's menu.



Note :

(i)
All the cloud cover input values (shown in **Fig. C-2**) has been reduced to a SINGLE-DIGIT "cloudiness index" value ranging from 0 to 9!

(ii)
Refer to **Fig. C-2**, the origin grid area [0, 0] previously have a value of '**41**', which is now transformed into a single-digit "cloudiness index" value = **4** !!!

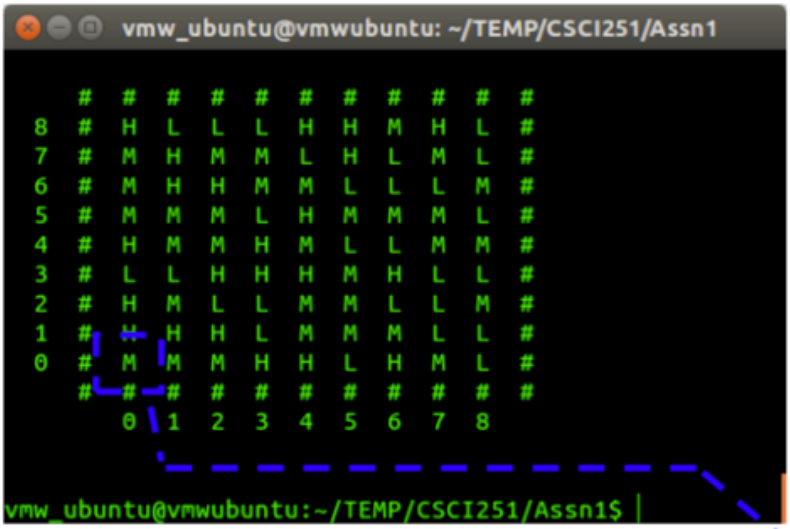
Fig. C-3

a) To generate the cloudiness index values for each grid area, all original cloud cover input values (for each grid area shown in **fig. C-2**) should be processed as follows:

	For each grid area [x, y], if original cloud cover value falls within the range ...	Then the corresponding "cloudiness index" value is :
1	0 <= value < 10	0
2	10 <= value < 20	1
3	20 <= value < 30	2
4	30 <= value < 40	3
5	40 <= value < 50	4
6	50 <= value < 60	5
7	60 <= value < 70	6
8	70 <= value < 80	7
9	80 <= value < 90	8
10	90 <= value < 100	9

Processing requirements - display cloud coverage map (LMH symbols)

Figure C-4 below illustrates the actual display formatting requirements when user selects the "Display Cloud Coverage Map (LMH symbols)" option from your program's menu.



Note :

(i)
All the cloud cover input values (shown in **Fig. C-2**) has been reduced to a SINGLE-CHAR "symbol" with values that is either 'L' / 'M' / 'H' !

(ii)
Refer to **Fig. C-2**, the origin grid area [0, 0] previously have a value of '**41**', which is now transformed into a single-char cloudiness symbol value = **M** !!!

Fig. C-4

a) To generate the LMH cloudiness symbol values for each grid area, all original cloud cover input values (for each grid area shown in **fig. C-2**) should be processed as follows:

	For each grid area [x, y], if original cloud cover value falls within the range ...	Then the corresponding "cloudiness symbol" value is :
1	0 <= value < 35	L (Low)
2	35 <= value < 65	M (Mid)
3	65 <= value < 100	H (High)

Atmospheric Pressure Input Data & Output Requirements

Figure D-1 below provides a sample of the **actual contents** within the input file (e.g. `pressure.txt`), storing the atmospheric pressure input data, for each grid area in the map.

Note : due to lack of "vertical space" on this page, the entire content is split across Figures **D-1a**, **D-1b** and **D-1c**. In reality, the data in all 3 figures are combined and stored in a single input file!

Fig. D-1a

```
[0, 0]-36
[0, 1]-32
[0, 2]-38
[0, 3]-64
[0, 4]-9
[0, 5]-62
[0, 6]-56
[0, 7]-41
[0, 8]-56
[1, 0]-24
[1, 1]-65
[1, 2]-27
[1, 3]-27
[1, 4]-85
[1, 5]-83
[1, 6]-6
[1, 7]-38
[1, 8]-82
[2, 0]-72
[2, 1]-42
[2, 2]-17
[2, 3]-15
[2, 4]-79
[2, 5]-11
[2, 6]-10
[2, 7]-47
[2, 8]-39
[3, 0]-53
[3, 1]-86
[3, 2]-33
[3, 3]-13
[3, 4]-15
[3, 5]-59
[3, 6]-16
[3, 7]-39
[3, 8]-38
```

Fig. D-1b

```
[4, 0]-15
[4, 1]-54
[4, 2]-58
[4, 3]-84
[4, 4]-47
[4, 5]-78
[4, 6]-68
[4, 7]-69
[4, 8]-25
[5, 0]-52
[5, 1]-36
[5, 2]-35
[5, 3]-40
[5, 4]-65
[5, 5]-40
[5, 6]-83
[5, 7]-50
[5, 8]-23
[6, 0]-27
[6, 1]-59
[6, 2]-42
[6, 3]-42
[6, 4]-63
[6, 5]-51
[6, 6]-5
[6, 7]-90
[6, 8]-83
[7, 0]-36
[7, 1]-53
[7, 2]-63
[7, 3]-77
[7, 4]-78
[7, 5]-50
[7, 6]-86
[7, 7]-24
[7, 8]-33
```

Interpretation of **each** line :

Data item **#1** :
`[x, y]` => the grid indices of the "lower-left" corner of a grid area

Delimiter char between data items **#1** and **#2** => `'-'`

Data item **#2** :
`15` => the "next day" forecast of atmospheric pressure for the grid area.

Max value : **99**
- means the grid area will experience extremely high pressure

Min value : **0**
- means the grid area will experience no pressure!

Fig. D-1c

```
[8, 0]-70
[8, 1]-32
[8, 2]-71
[8, 3]-18
[8, 4]-29
[8, 5]-73
[8, 6]-40
[8, 7]-80
[8, 8]-78
```

To aid in the visualization and understanding of how the atmospheric pressure input data will look like, in a 2D graphical format, please refer to the figure D-2 below.

Referring to the FIRST LINE shown in earlier **Fig. D-1a**, the grid area [0, 0] has a pressure value of 36, which is shown here.

Referring to the FIRST LINE shown in earlier **Fig. D-1b**, the grid area [0, 4] has a pressure value of 15, which is shown here.

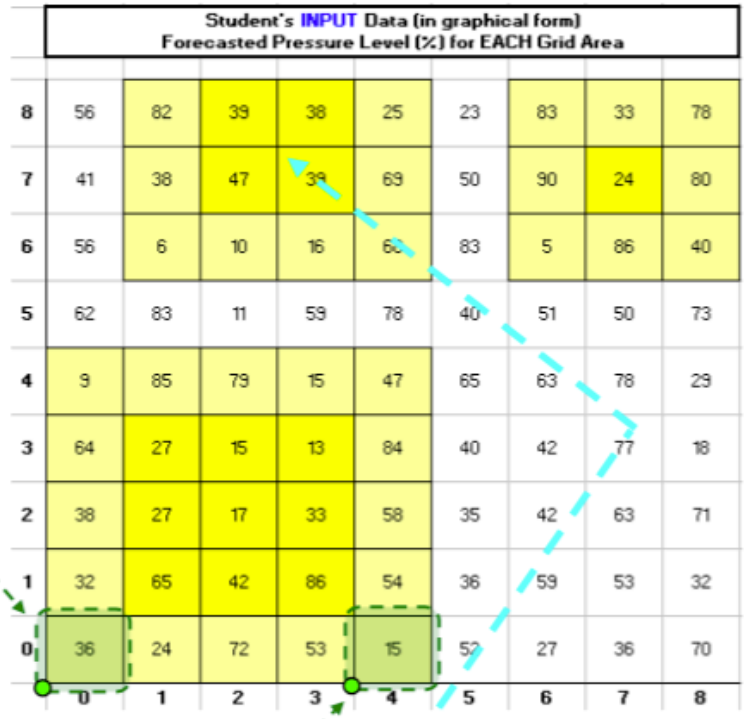


Fig. D-2

Note :

a) Bright Yellow grid areas indicate the ACTUAL Grid Area occupied by a particular city. For example, in the **fig. D-2** above, the city named "Mid_City" occupies the following grid areas : [2, 7], [2, 8], [3, 7], [3, 8]

b) Light Yellow grid areas indicate the grid areas surrounding the perimeter of EACH city. For example, in **fig. D-2** above, the city named "Mid_City" has the following grid areas surrounding its perimeter : [1, 6], [1, 7], [1, 8], [2, 6], [3, 6], [4, 6], [4, 7], [4, 8]

- The grid areas [1, 9], [2, 9], [3, 9], [4, 9] are not shown as they are **BEYOND the upper limits** of the vertical `GridY_IdxRange` ! As a result, they are ***not included*** in the **AP** computation (see below) as well ...

c) For each city, the AVERAGE PRESSURE (AP) value is derived, by the following :

AP = SUM (cloud cover values of city + surrounding grid areas) / Total # of grid areas

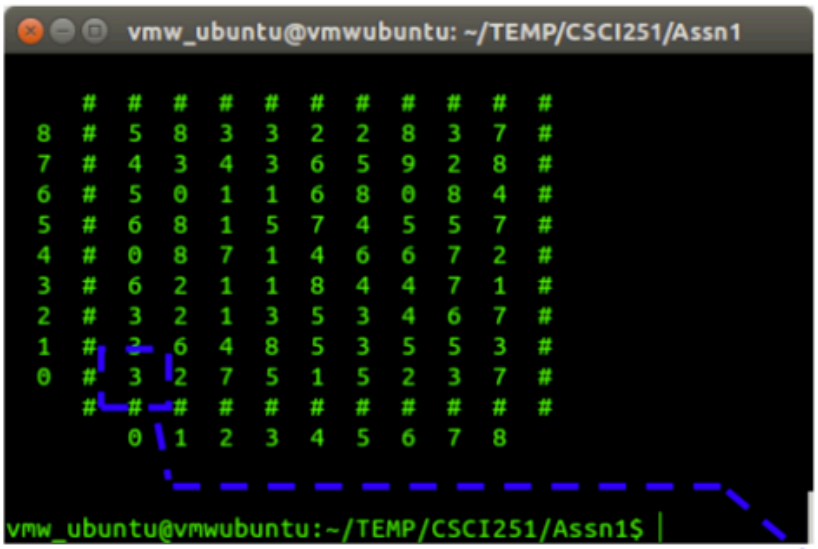
For example, for the city named "Mid_City" :

AP = ((47 + 39 + 39 + 38) + (6 + 38 + 82 + 10 + 16 + 68 + 69 + 25)) / 12

= **39.75**

Processing requirements - display atmospheric pressure map (pressure index)

Figure D-3 below illustrates the actual display formatting requirements when user selects the "Display Atmospheric Pressure Map (pressure index)" option from your program's menu.



Note :

(i)
All the cloud cover input values (shown in **Fig. D-2**) has been reduced to a SINGLE-DIGIT "pressure index" value ranging from 0 to 9!

(ii)
Refer to **Fig. D-2**, the origin grid area [0, 0] previously have a value of '**36**', which is now transformed into a single-digit pressure index value = 3 !!!

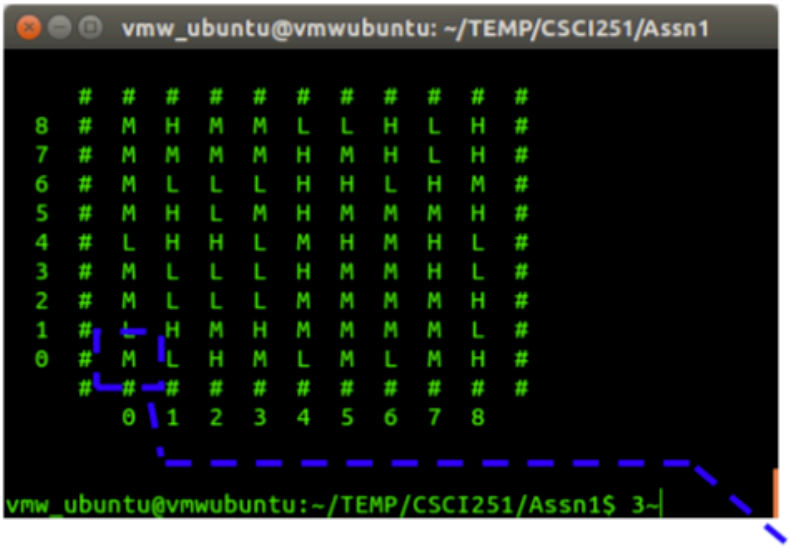
Fig. D-3

a) To generate the pressure index values for each grid area, all original atmospheric pressure input values (for each grid area shown in **fig. D-2**) should be processed as follows:

	For each grid area [x, y], if original pressure value falls within the range ...	Then the corresponding "pressure index" value is :
1	0 <= value < 10	0
2	10 <= value < 20	1
3	20 <= value < 30	2
4	30 <= value < 40	3
5	40 <= value < 50	4
6	50 <= value < 60	5
7	60 <= value < 70	6
8	70 <= value < 80	7
9	80 <= value < 90	8
10	90 <= value < 100	9

Processing requirements - display atmospheric pressure map (LHM)

Figure D-4 below illustrates the actual display formatting requirements when user selects the " Display Atmospheric Pressure Map (LMH symbols)" option from your program's menu.



Note :

(i)
All the pressure input values (shown in **Fig. D-2**) has been reduced to a SINGLE-CHAR "symbol" with values that is either 'L' / 'M' / 'H' !

(ii)
Refer to **Fig. D-2**, the origin grid area [0, 0] previously have a value of '**36**', which is now transformed into a single-char pressure symbol value = **M** !!!

Fig. D-4

a) To generate the LMH pressure symbol values for each grid area, all original atmospheric pressure input values (for each grid area shown in **fig. D-2**) should be processed as follows:

	For each grid area [x, y], if original pressure value falls within the range ...	Then the corresponding "pressure symbol" value is :
1	`0 <= value < 35`	`L (Low)`
2	`35 <= value < 65`	`M (Mid)`
3	`65 <= value < 100`	`H (High)`

Weather Forecast Summary Report

Figure E-1 below illustrates the actual display requirements when user selects the "Show Weather Forecast Summary Report" option from your program's menu.

Referring to **Appendix B**, **Fig B-1**, ***for each city*** described in the sample input file (citylocation.txt), display the city's **name**, **id**, **ACC**, **AP**, Probability of Rain and its corresponding ASCII graphics!

```
Weather Forecast Summary Report
-------------------------------
City Name : Big_City
City ID   : 3
Ave. Cloud Cover (ACC)    : 56.60 (M)
Ave. Pressure    (AP)     : 43.60 (M)
Probability of Rain (%)   : 50.00
~~~~
~~~~~
    \

City Name : Mid_City
City ID   : 2
Ave. Cloud Cover (ACC)    : 50.83 (M)
Ave. Pressure    (AP)     : 39.75 (M)
Probability of Rain (%)   : 50.00
~~~~
~~~~~
    \

City Name : Small_City
City ID   : 1
Ave. Cloud Cover (ACC)    : 29.56 (L)
Ave. Pressure    (AP)     : 56.67 (M)
Probability of Rain (%)   : 40.00
~~~~
~~~~~
```

Fig. E-1

Note :

(i)
For Ave. Cloud Cover (ACC) computation, please refer to Note: part c) section (below **Fig. C-2**) in **Appendix C**

(ii)
For Ave. Pressure (AP) computation, please refer to Note: part c) section (below **Fig. D-2**) in **Appendix D**

(iii)
The L/M/H symbol for ACC is computed based on table (below **Fig. C-4**) in **Appendix C**

(iv)
The L/M/H symbol for AP is computed based on table (below **Fig. D-4**) in **Appendix D**

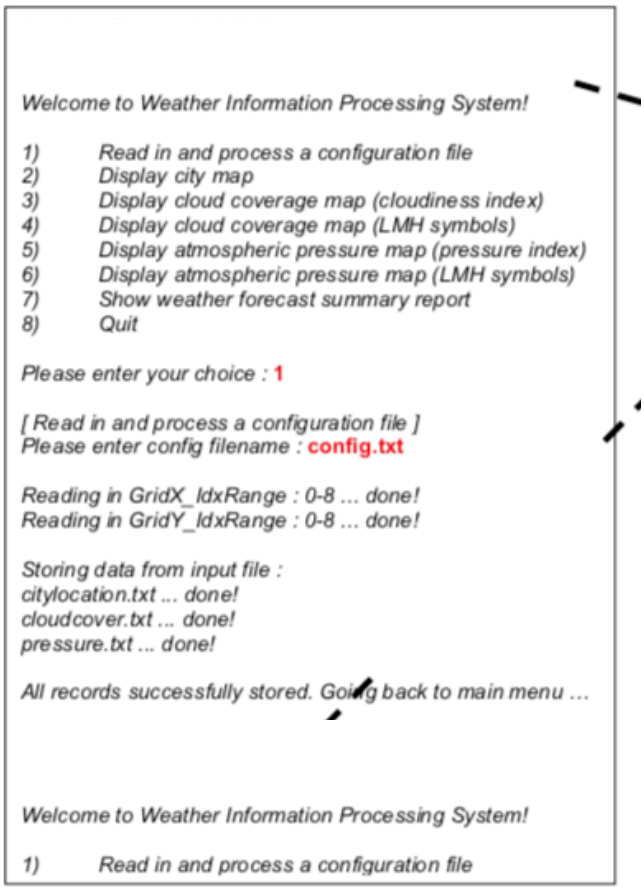
Determination of Probability of Rain (%) value

With reference to earlier **fig. E-1**, for each city, note down their L/M/H symbols for ACC and AP respectively. Based on the combination of ACC and AP symbols, lookup the respective Probability of Rain (%) value from the table below.

	City's forecasted ...		Estd. Probability of Rain (%)	Associated ASCII graphics (to be displayed)
	Ave. Pressure (**AP**) value	Ave. Cloud Cover (**ACC**) value		
1	L	H	90	~~~~ ~~~~~ \\\\\
2	L	M	80	~~~~ ~~~~~ \\\\
3	L	L	70	~~~~ ~~~~~ \\\
4	M	H	60	~~~~ ~~~~~ \\
5	M	M	50	~~~~ ~~~~~ \
6	M	L	40	~~~~ ~~~~~
7	H	H	30	~~~ ~~~~
8	H	M	20	~~ ~~~
9	H	L	10	~ ~~

Note : Air will always flow from areas of High pressure, to areas of Low pressure, and this is manifested as "wind". If the city is forecasted to have Low pressure, then wind (bringing along additional clouds & moisture), is predicted to flow into that city on the next day!

Main Menu Requirements



Note : if user selects option ...

2) Display City Map as described by **Appendix B**, **Fig. B-3**
3) Display Cloud Coverage Map (cloudiness index) as described by **Appendix C**, **Fig. C-3**
4) Display Cloud Coverage Map (LMH symbols) as described by **Appendix C**, **Fig. C-4**
5) Display Atmospheric Pressure Map (pressure index) as described by **Appendix D**, **Fig. D-3**
6) Display Atmospheric Pressure Map (LMH symbols) as described by **Appendix D**, **Fig. D-3**
7) Show Weather Forecast Summary Report as described by **Appendix E**, **Fig. E-1**

For options 2) to 7), **always** prompt user "*Press <enter> to go back to main menu ...*",
so that when user is done viewing the output, he can hit <enter> to continue