CSCI251 Advanced Programming

Assignment 1

Aim

The objectives of this assignment includes:
- Learning about procedural programming, control structures, arrays, dynamic memory allocation and handling input/output
- Apply the concepts learnt by developing a Weather Information Processing System

Background

Your software company, Virtual Solutions, has just won a $1m contract to design a build a prototype Weather Information Processing System (WIPS).

For this assignment, you take on the role of a software architect. You are supplied with sample input weather data (kindly provided by the meteorological station), and you need to develop a program that does the following:

1) read in and process a configuration file
2) display city map
3) display cloud coverage map (cloudiness index)
4) display cloud coverage map (LMH symbols)
5) display atmospheric pressure map (pressure index)
6) display atmospheric pressure map (LMH symbols)
7) show weather forecast summary report

The program should be compiled as 'csci251_a1.exe' and run in Ubuntu 14.04 (Linux OS). The next section describes the requirements for the program.

Task Requirements

A) Upon startup, the program should prompt user to enter a 'configuration' filename. The program then proceeds to read the contents (of the entered filename) to further initialize its own program parameters / data structures. Please refer to **Appendix A**, which provides details on the configuration file and its usage.

B) The meteorological station has adopted a map-grid coordinate system upon which it overlays all other weather data. With regards to display city map option, please refer to **Appendix B**, which elaborates on this coordinate system, the unit representation, the relative positioning of different cities and its display requirements.

C) For the display requirements of cloud coverage (cloudiness index), AND cloud coverage (LMH symbols) (i.e. options 3) & 4)), please refer to **Appendix C**, which will discuss about the interpretation of cloud cover input data, its subsequent processing algorithm and output format requirements.

CSCI251 Advanced Programming

D) For the display requirements of <u>atmospheric pressure (pressure index)</u> AND <u>atmospheric pressure (LMH symbols)</u> (i.e. options 5) & 6)), please refer to **Appendix D**, which will discuss about the interpretation of pressure input data, its subsequent processing algorithm and output format requirements.

E) For the display requirements of weather forecast summary report, please refer to **Appendix E**, which will provides details about the kind of summarized weather data to be displayed, processing algorithm to compute the probability of rain and its graphical display format.

F) Refer to **Appendix F**, for a description of the program's main menu requirements

G) Once the program is completed and tested to be working successfully, you are encouraged to add on "new features" to the program that you feel are relevant to the problem. Additional marks may be awarded subject to the relevancy and correctness of the new functionalities. (Note : the additional features will only be considered IF the program has correctly fulfilled all the basic requirements elaborated in the earlier sections!)

H) You are **not allowed** to declare your own C++ classes for this program, as this is a purely procedural (not Object Oriented) programming assignment!

I) You are to use only C++ language to develop your program. There is no restriction on the IDE as long as your source files can be **compiled by g++ compiler** (that comes installed in Tutor's Ubuntu Linux) and <u>executed in the Ubuntu terminal shell environment</u>.

Deliverables

1) The deliverables include the following:

   a) The actual working C++ program (soft copy), ***with comments on each file, function or block of code*** to help the tutor understand its purpose.

   b) A softcopy word document that elaborates on:
   - (Interpreted) requirements of the program
   - Diagram / Illustrations of program design
   - Summary of implementation of each module in your program
   - Reflections on program development (e.g. assumptions made, difficulties faced, what could have been done better, possible enhancements in future, ***what have you learnt***, etc)

   c) A program demo/software testing during lab session. You must be prepared to perform certain tasks / answer any questions posed by the tutor.

2) IMPT: Please **follow closely**, to the submission instructions in **Appendix G**, which contains details about what to submit, file naming conventions, when to submit, where to submit, etc.

3) The software demo / testing will be held during lab session where you are supposed to submit your assignment. Some time will be allocated for you to present / demonstrate your program's capabilities during the session.

Grading

Student's deliverable will be graded according to the following criteria:

(i) Program fulfills all the basic requirements stipulated by the assignment

(ii) Successful demonstration of a working program, clarity of explanation / presentation and satisfactory answers provided during Q & A session.

(iii) Additional effort (e.g. enhancing the program with *relevant* features over and above task requirements, impressive, 'killer' presentation)

(iv) After the submission of deliverables, students will be required undergo a software testing process (to determine the correctness and fulfillment of software requirements.) Further instructions will be given by the Tutor during the subsequent respective labs. Please pay attention as failure to adhere to instructions will result in deduction of marks.

Tutor's note:
In the real working world, satisfactory completion of your tasks is no longer enough. The capability, efficiency and robustness of your system to operate under different testing conditions, and the ability to add value, communicate and/or demonstrate your ideas with clarity is just as important as correct functioning of your program. The grading criteria is set to imitate such requirements on a 'smaller scale'.

CSCI251 Advanced Programming

APPENDIX A

Processing Configuration File Info

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!

Regarding the interpretation of each input files' data format, please refer to subsequent **Appendices**.

APPENDIX B

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.

```
[1, 1]-3-Big_City
[1, 2]-3-Big_City
[1, 3]-3-Big_City
[2, 1]-3-Big_City
[2, 2]-3-Big_City
[2, 3]-3-Big_City
[2, 7]-2-Mid_City
[2, 8]-2-Mid_City
[3, 1]-3-Big_City
[3, 2]-3-Big_City
[3, 3]-3-Big_City
[3, 7]-2-Mid_City
[3, 8]-2-Mid_City
[7, 7]-1-Small_City
```

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** :
3 => the city id (to be displayed later on map)

Delimiter char between data items **#2** and **#3** => **'-'**

Data item **#3** :
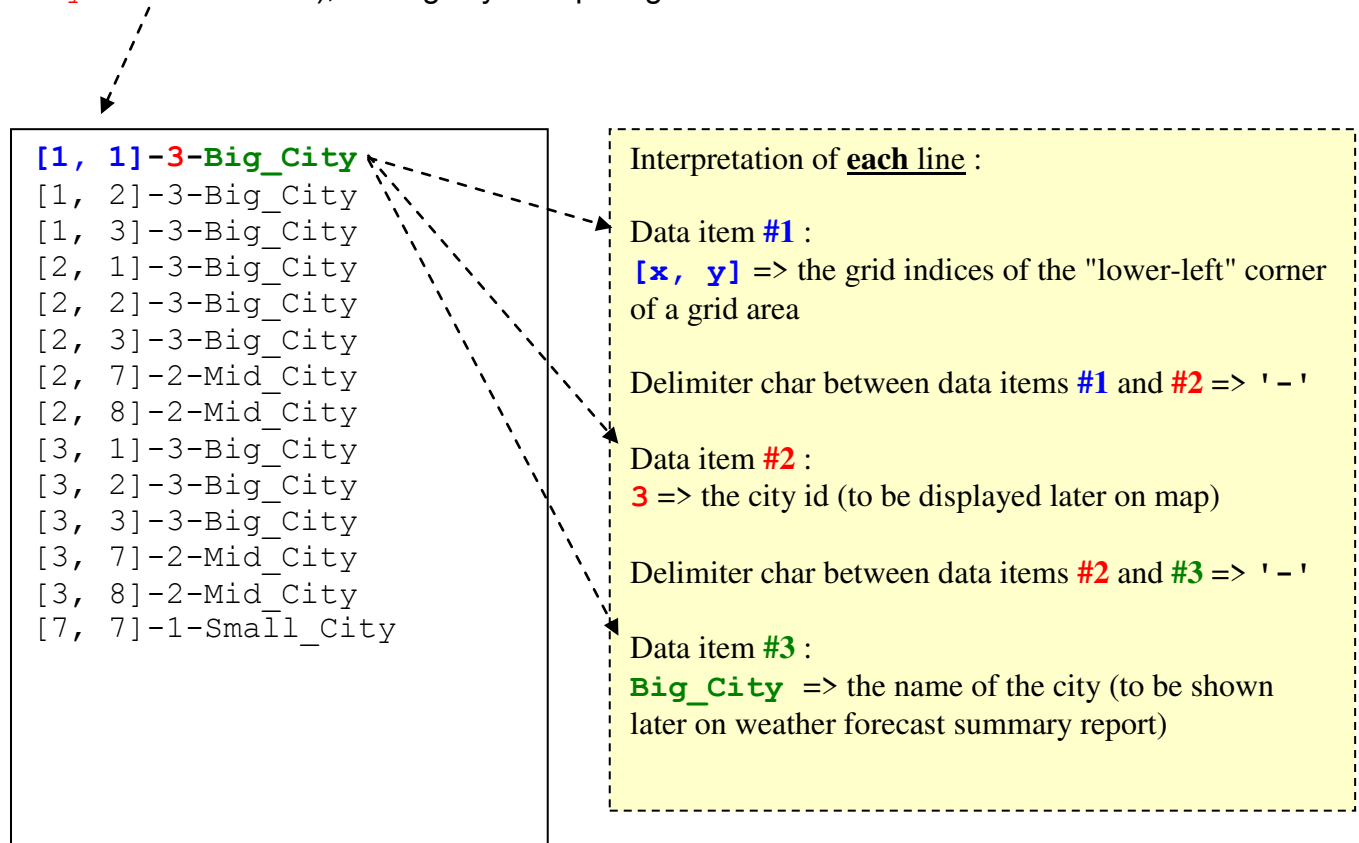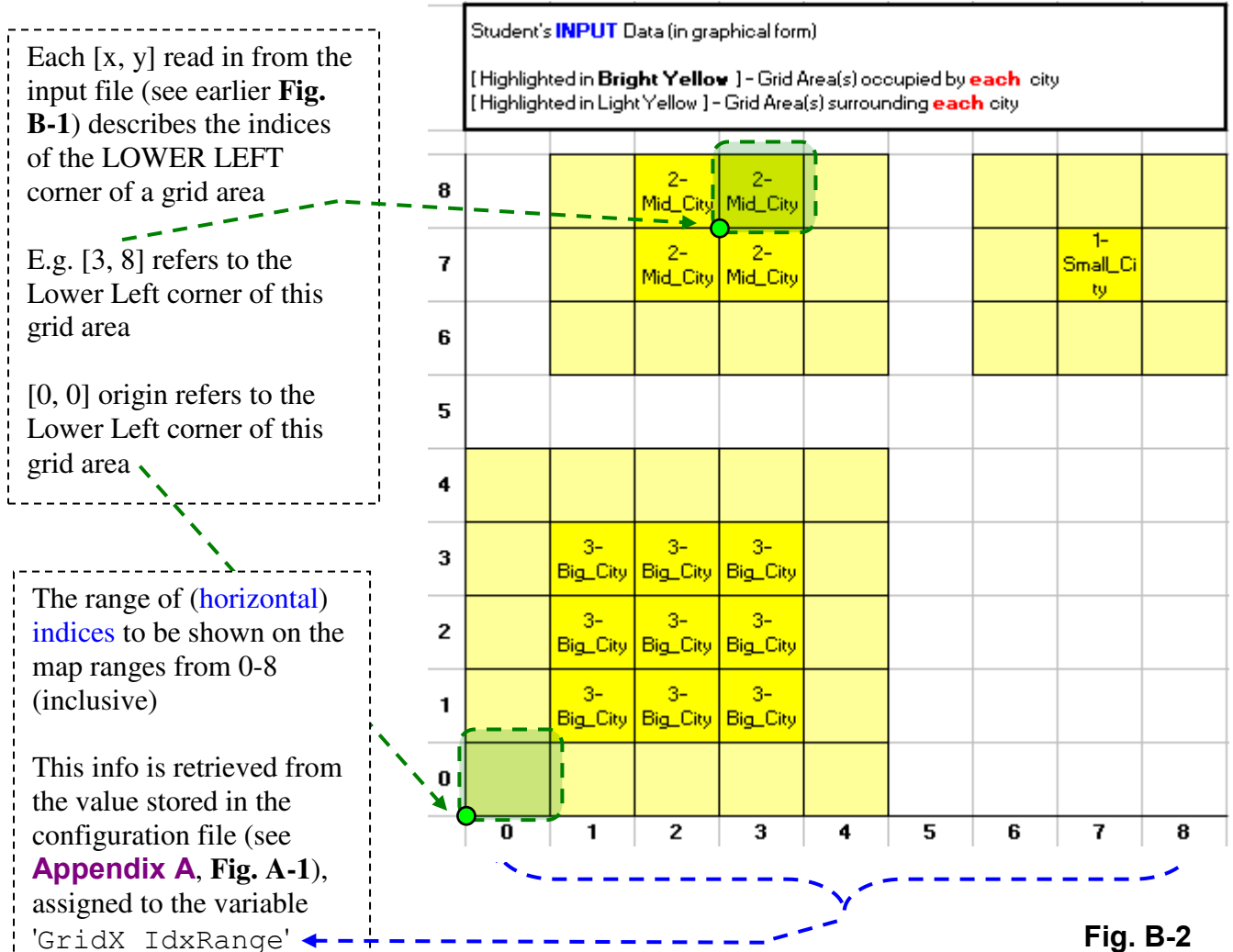**Big_City** => the name of the city (to be shown later on weather forecast summary report)

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

Student's **INPUT** Data (in graphical form)

[Highlighted in **Bright Yellow**] – Grid Area(s) occupied by **each** city
[Highlighted in Light Yellow] – Grid Area(s) surrounding **each** city

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 !

CSCI251 Advanced Programming

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 <u>actual display formatting requirements</u> when user selects the "Display City Map" option from your program's menu.
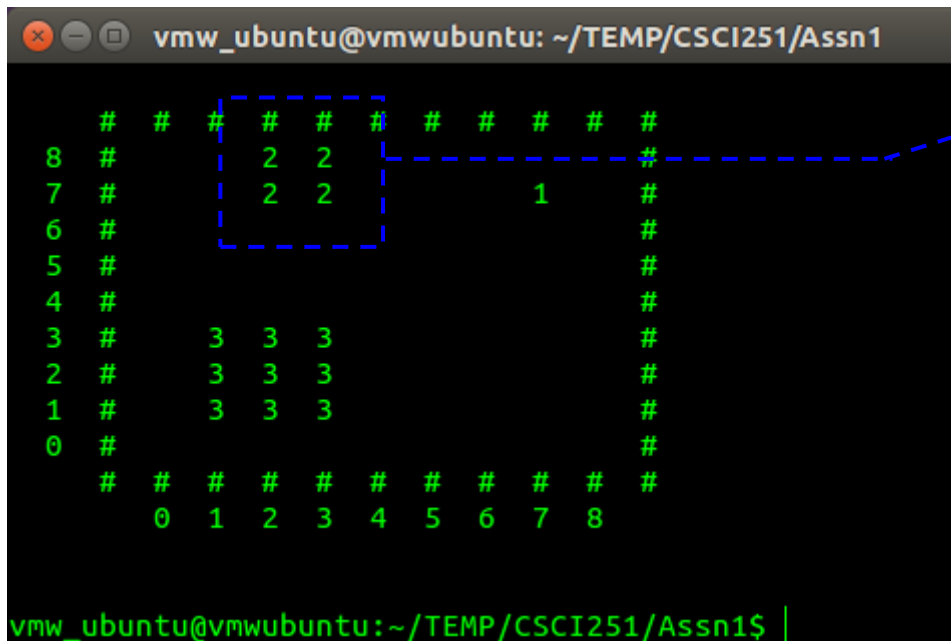
Fig. B-3

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 <u>can change</u> 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!

APPENDIX C

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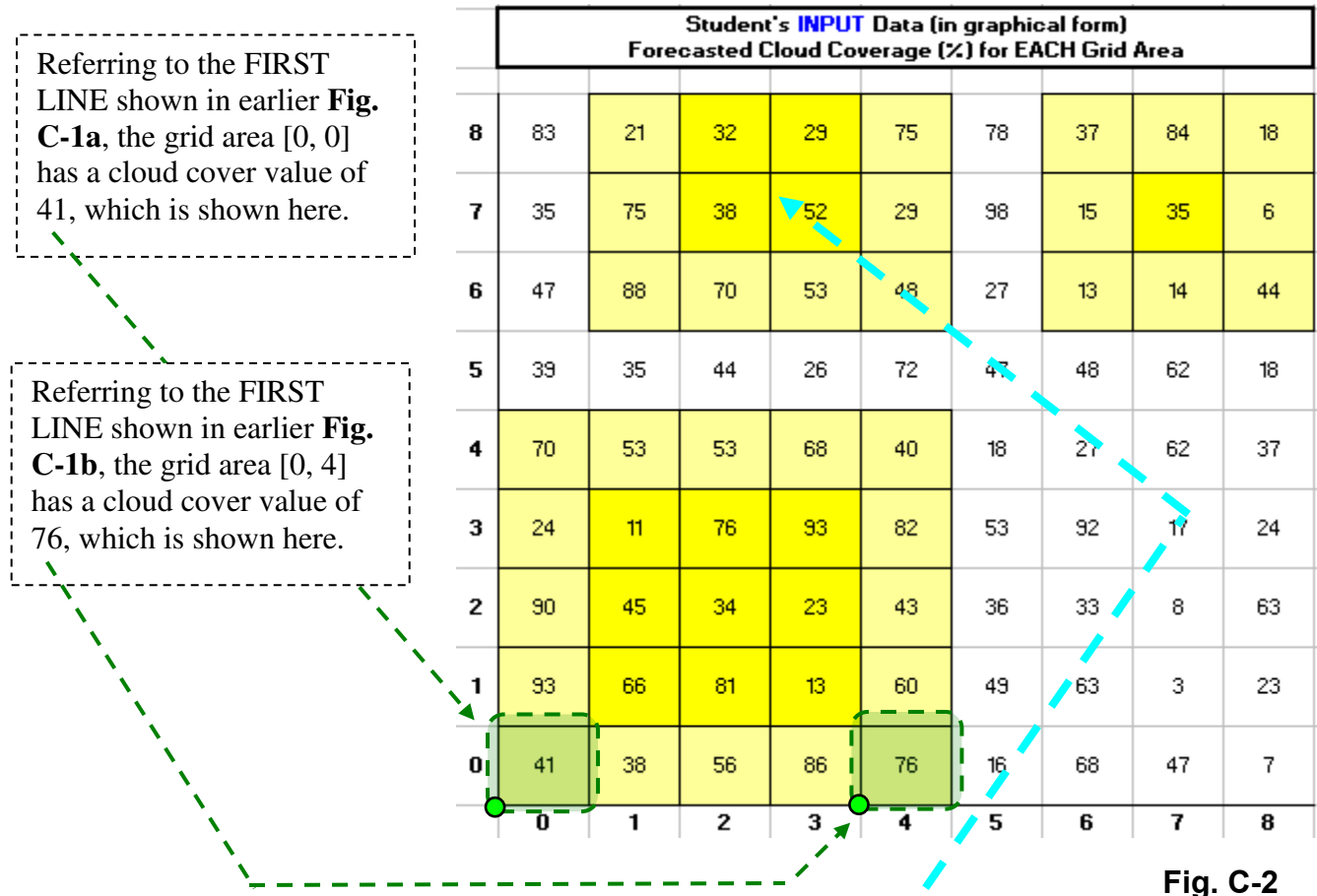
Min value : **0**
- means not a single wisp of cloud - "clear blue skies" for the grid area

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.
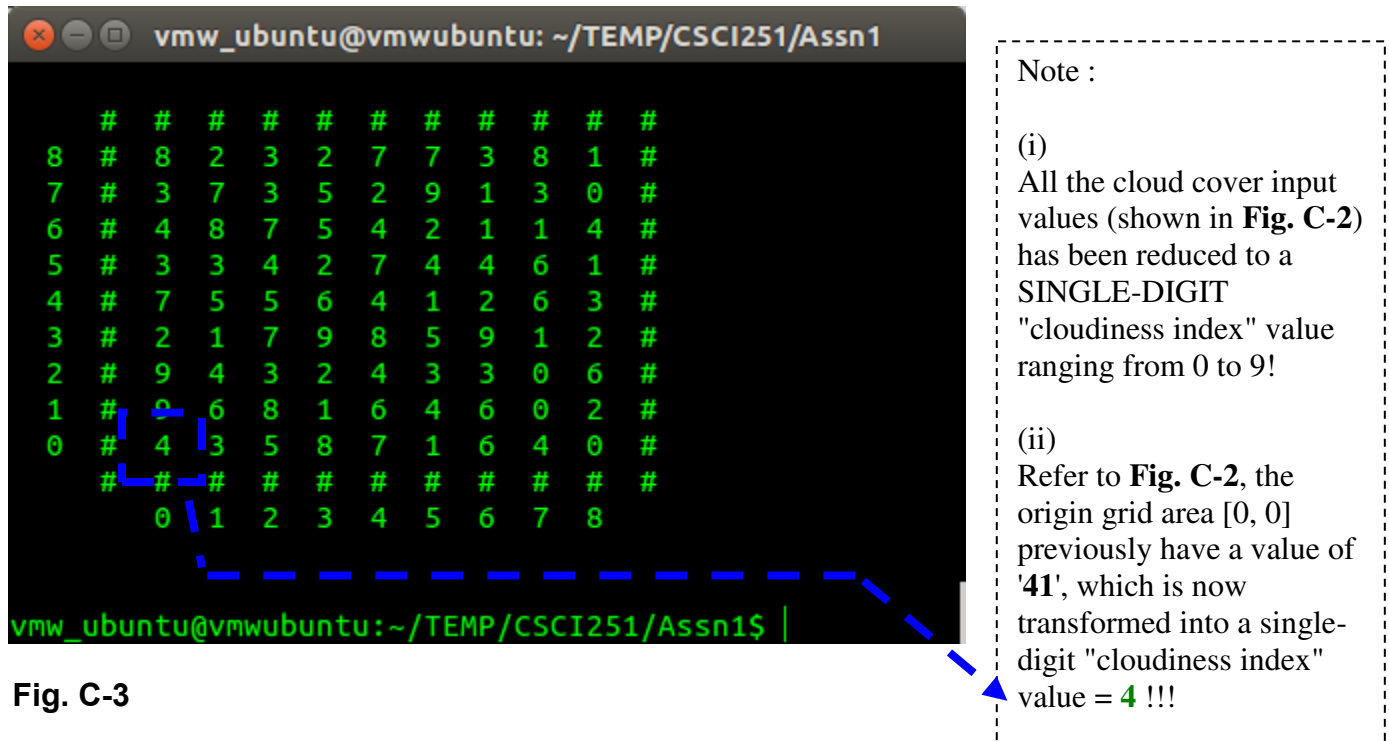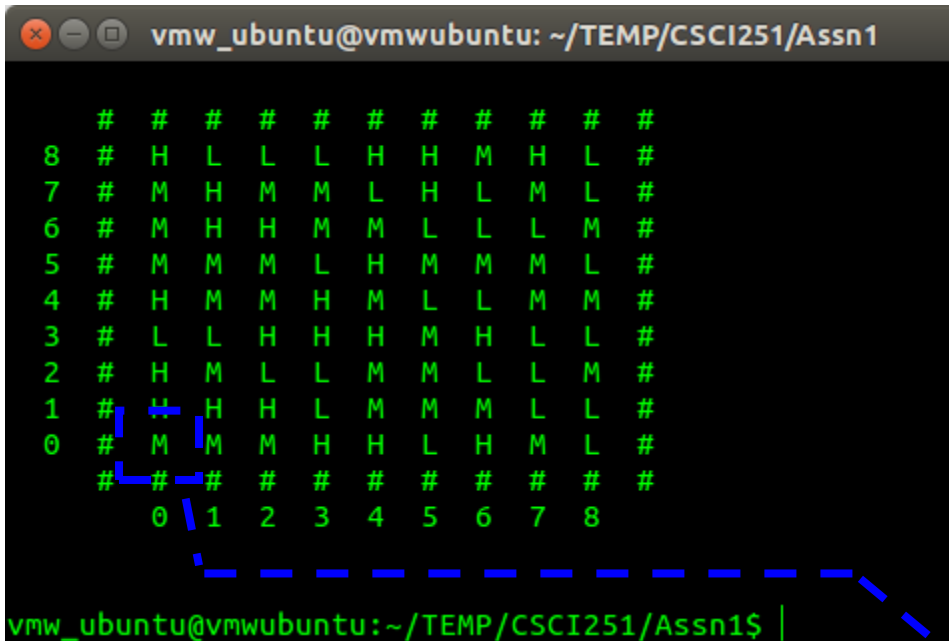

Fig. C-3

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** !!!

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.

Fig. C-4

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** !!!

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)

APPENDIX D

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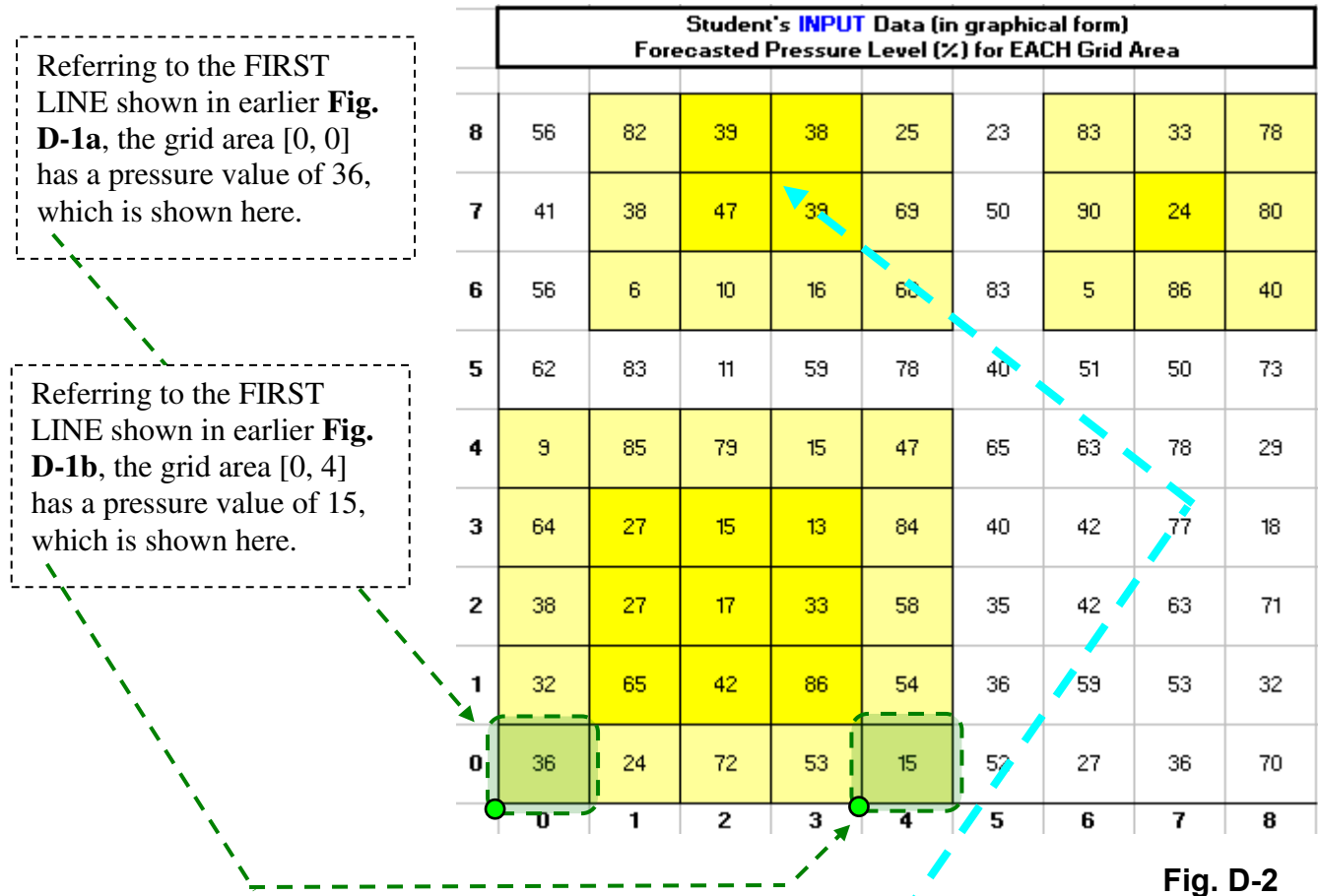
Min value : **0**
- means the grid area will experience no pressure!

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.

	Student's INPUT Data (in graphical form) Forecasted Pressure Level (%) for EACH Grid Area								
8	56	82	39	38	25	23	83	33	78
7	41	38	47	39	69	50	90	24	80
6	56	6	10	16	68	83	5	86	40
5	62	83	11	59	78	40	51	50	73
4	9	85	79	15	47	65	63	78	29
3	64	27	15	13	84	40	42	77	18
2	38	27	17	33	58	35	42	63	71
1	32	65	42	86	54	36	59	53	32
0	36	24	72	53	15	52	27	36	70
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**

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.
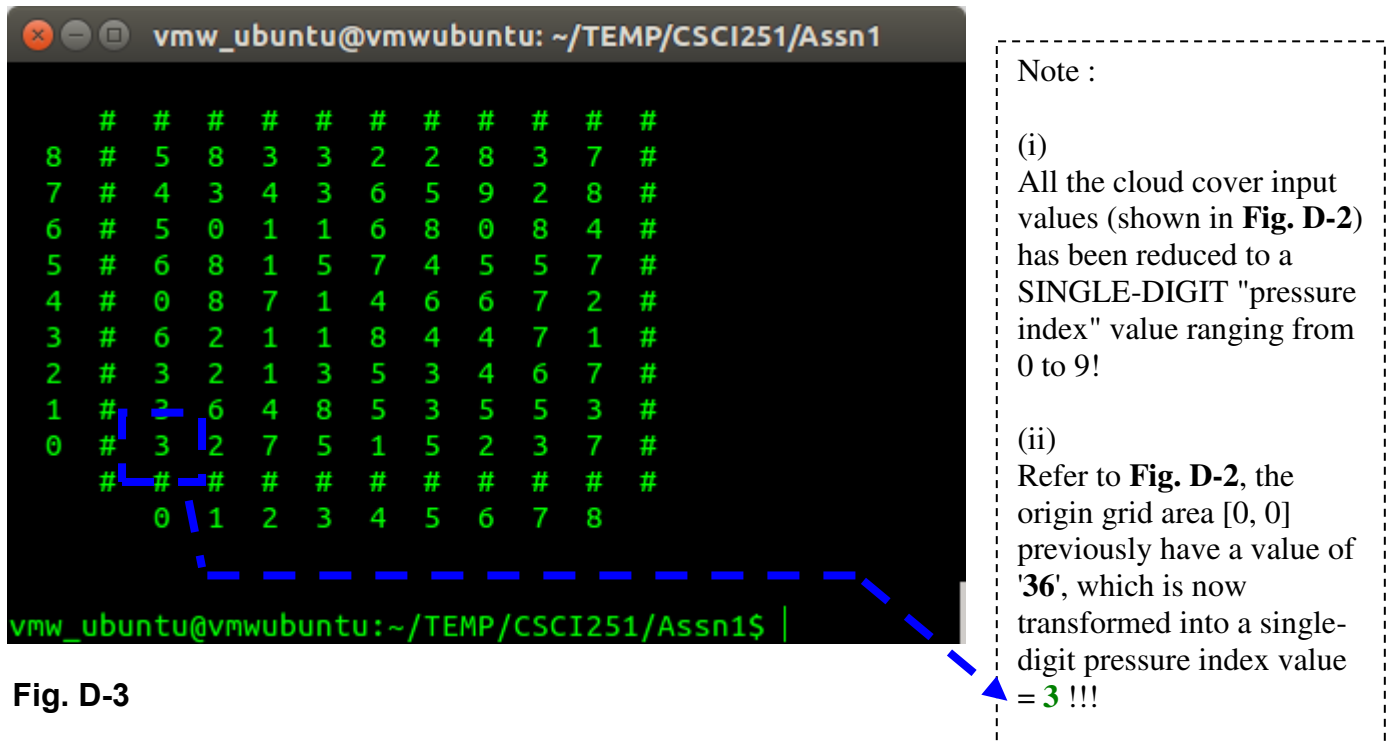

Fig. D-3

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** !!!

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 (LMH symbols)

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.

Fig. D-4
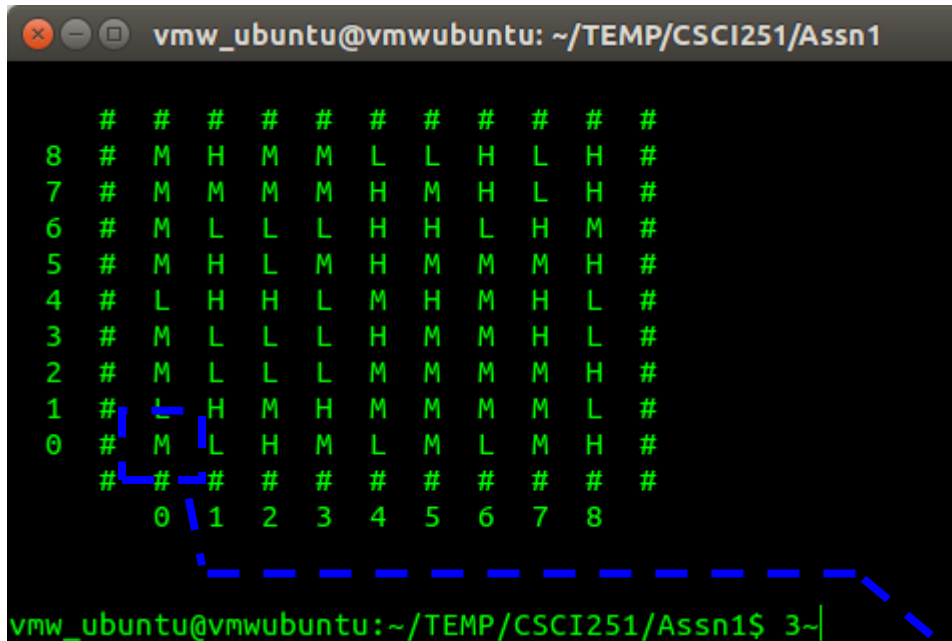
Note :

(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** !!!

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:

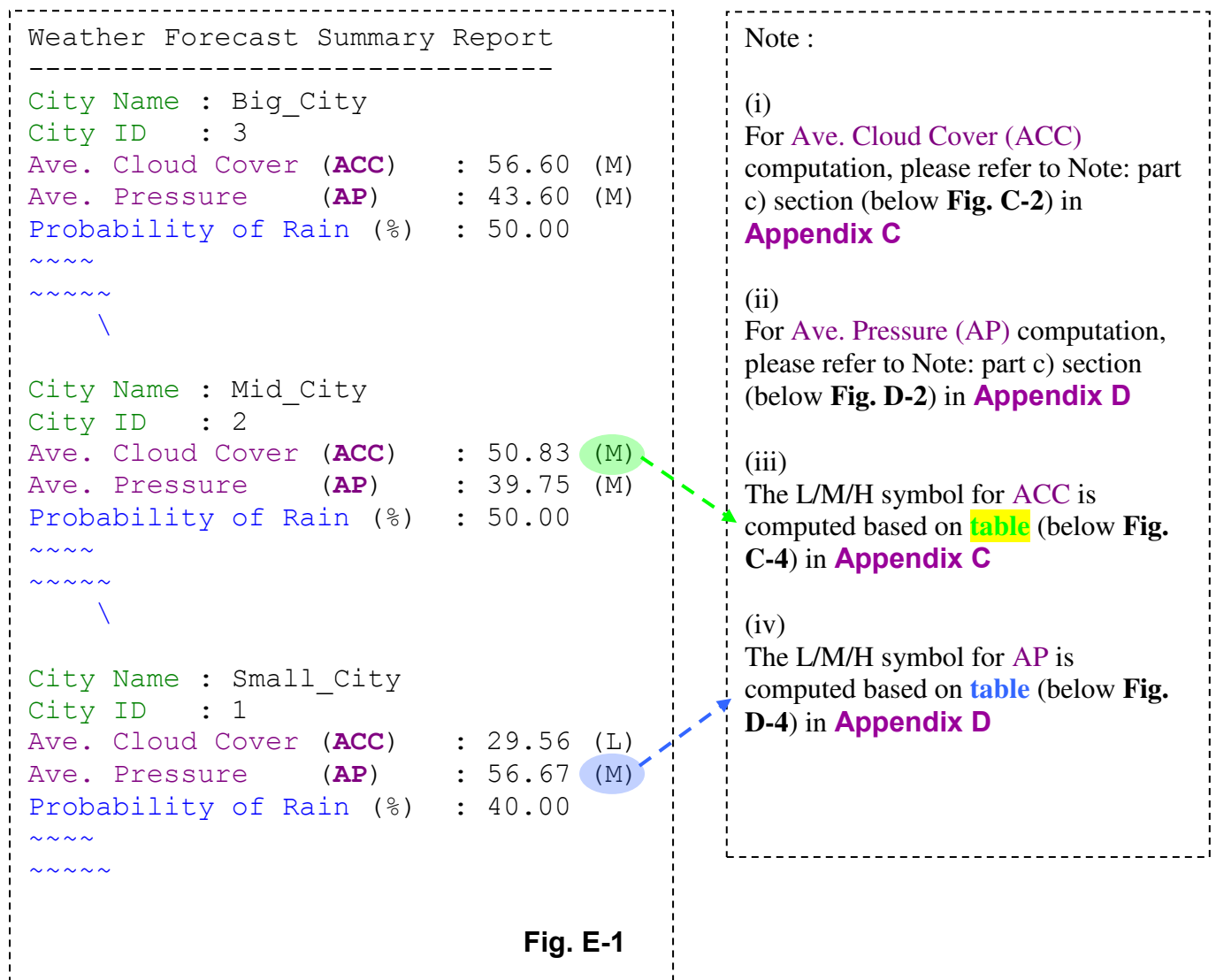|	For each grid area [x, y], if original pressure value falls within the range ...	Then the corresponding "pressure symbol" value is :
1	0 <= value < 35	L (Low)
2	35 <= value < 65	M (Mid)
3	65 <= value < 100	H (High)

APPENDIX E

(Weather Forecast Summary Report)

Figure E-1 below illustrates the <u>actual display requirements</u> 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!

APPENDIX F

(Main Menu Requirements)

Student ID   : 1234567
Student Name : Tan Ah Meng Elvis
\---
Welcome to Weather Information Processing System!

1)   Read in and process a configuration file
2)   Display city map
3)   Display cloud coverage map (cloudiness index)
4)   Display cloud coverage map (LMH symbols)
5)   Display atmospheric pressure map (pressure index)
6)   Display atmospheric pressure map (LMH symbols)
7)   Show weather forecast summary report
8)   Quit

Please enter your choice : **1**

[Read in and process a configuration file]
Please enter config filename : **config.txt**

Reading in GridX_IdxRange : 0-8 ... done!
Reading in GridY_IdxRange : 0-8 ... done!

Storing data from input file :
citylocation.txt ... done!
cloudcover.txt ... done!
pressure.txt ... done!

All records successfully stored. Going back to main menu …

Student ID   : 1234567
Student Name : Tan Ah Meng Elvis
\--
Welcome to Weather Information Processing System!

1)   Read in and process a configuration file

Impt !! Please include your:
1. UOW Student ID
2. Student Name

every time you display your Main Menu. Marks will be deducted if the required info is not shown.

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

---

CSCI251 Advanced Programming

APPENDIX G

Submission Instructions (V. IMPT!!)

1) Deliverables

    a) All submissions should be in **softcopy**, unless otherwise instructed

    b) For the actual files to be submitted, you typically need to include the following:

- word document report (e.g. ***.doc**), save as MS Word 97-2003 format

- the source file(s), (e.g. ***.h**, ***.o**, or ***.cpp** files)

- the executable file, compile into an executable filename with ***.exe**
    (e.g. csci251_a1.**exe**)

2) How to package

Compress all your assignment files into a single zip file. Please use the following naming format :

<div align="center">

<FT/PT>_Assn1_<Stud. No.>_<Name>.zip

</div>

Example : **FT_Assn1_1234567_JohnDoeAnderson. zip**

- **<FT/PT>** Use "**FT**" for **F**ull-**T**ime student, "**PT**" if you are **P**art-**T**ime student

- Assn**1** if you are submitting assignment **1**, Assn**2** if submitting assignment **2** etc.

- **<Stud. No.>** refers to your UOW student number (e.g. 1234567)

- **<Name>** refers to your UOW registered name (e.g. JohnDoeAnderson)

3) Where to submit

Please submit your assignment via Moodle eLearning site.

IMPORTANT NOTE :

- To minimize the chances of encountering **UNFORSEEN SITUATIONS** (mentioned below), please do an <u>EARLY SUBMISSION</u> via Moodle.
- Prior to the relevant assignment deadline, you can upload (and replace) your assignment submissions in Moodle **as many times as you like**
- It is <u>your responsibility</u> to confirm that you have submitted the final (and correct version) of your deliverables to Moodle <u>before deadline</u>
- Any submission uploaded to Moodle after deadline will be considered late

In the event of UNFORSEEN SITUATIONS :

(E.g. 3 hrs prior to deadline, there is **proof of** unforseen events like Moodle site down, unable to upload assignment, undersea internet cable damaged by sea urchins, etc)

Please email your single zip file to your tutor at :

csci251@yahoo.com   for *FULL* TIME students

csci251@yahoo.com   for *PART* TIME students

In your email **subject** line, type in the following information :

<FT/PT> <assignment info> <**student number**> and <**name**>.

Example:

To   :   **tutor's email** (see above)

Subject   :   FT Assn**1 1234567 JohnDoeAnderson**

Note 1 :   The timestamp shown on tutor's email Inbox will be used to determine if the assignment is late or not.

Note 2 :   After email submission, your mailbox's **sent folder** would have a copy (record) of your sent email, please **do not delete** that copy !! It could be used to prove your timely submission, in case the Tutor did not receive your email!

4) When to submit

a) Depending on the time-table, a <u>software demo / testing for your assignment</u> may be scheduled during the:

- 3rd - 5th lab session for the semester (i.e. lab 3 - 5), for Full Time (**FT**) students
- 2nd - 4th lab session for the semester (i.e. lab 2 - 4), for Part Time (**PT**) students

Please consult your tutor for further details. Some time may be allocated for each student to present / explain your system design during the session.

b) Please refer to the following table on the different submission events and deadlines

Assignment #	Submission Deadline (latest by 2300 hrs), night before :		Software Demo / Testing (Tasks), during ...	Email Demo / Software Testing result files by :
	PT	**FT**		
1	Lab 2	Lab 3	Lab 2(PT), Lab 3(FT)	End of Lab 2(PT), Lab 3(FT)
2	Lab 3	Lab 4	Lab 3(PT), Lab 4(FT)	End of Lab 3(PT), Lab 4(FT)
3	Lab 4	Lab 5	Lab 4(PT), Lab 5(FT)	End of Lab 4(PT), Lab 5(FT)

Note: (PT) = Part Time Students, (FT) = Full Time Students !

c) **Example #1**, for Full Time (**FT**) students submitting Assignment **1** ...

- Check SimConnect for the date of **Lab 3** (**e.g. 26 / 08 / 2018**)
- Then, submit <u>Assn **1**</u> zip file to Moodle by **25 / 08 / 2018**, 2300 hrs - **night before Lab 3**
- Setup your <u>Assn **1**</u> program for software demo/testing <u>on the date</u> of : **Lab 3**
- Finish demo/testing tasks, email <u>Assn **1**</u> test result files <u>by end</u> of : **Lab 3**

d) **Example #2**, for Part Time (**PT**) students submitting Assignment **3** ...

- Check SimConnect for the date of **Lab 4** (**e.g. 23 / 08 / 2018**)
- Then, submit <u>Assn **3**</u> zip file to Moodle by **22 / 08 / 2018**, 2300 hrs - **night before Lab 4**
- Setup your <u>Assn **3**</u> program for software demo/testing <u>on the date</u> of : **Lab 4**
- Finish demo/testing tasks, email <u>Assn **3**</u> test result files <u>by end</u> of : **Lab 4**

5) Please help by paying attention to the following …

! VERY IMPORTANT !

PLEASE FOLLOW ALL THE GUIDELINES / REQUIREMENTS IN ALL ASSIGNMENT APPENDICES !!

PLEASE FOLLOW ALL THE SUBMISSION INSTRUCTIONS FROM **1** TO **4** !!

IF YOU ARE **NOT SURE**,

> PLEASE **CHECK WITH YOUR TUTOR** DURING LABS / LECTURES !
>
> OR ...
>
> PLEASE EMAIL YOUR ENQUIRIES TO YOUR TUTOR !

MARKS WILL BE DEDUCTED IF YOU FAIL TO FOLLOW INSTRUCTIONS !!

Examples of marks deduction :

- Your deliverables / zip file does not follow naming convention
- You have no email subject / or do not following naming convention
- Your email address / content does not include your name/identity (i.e. tutor cannot easily identify sender)

- Wrong naming or **misleading information** given
  (e.g. putting "Assn2" in email subject, when you are submitting "Assn1")
  (e.g. naming "Assn1" in your zip file, but inside contains Assn2 files)

- You email to the WRONG tutor
- Your submission cannot be downloaded and unzipped
- Your program cannot be re-compiled and/or executable file cannot run
- Your report / testing files cannot be opened by Microsoft Word / Excel 2003
- You did not submit / incomplete submission of software demo / testing files
- etc

6) Re-submission administration

After the deadline, (on case-by-case basis), some students / grp may be granted an opportunity for an un-official resubmission by the tutor. If this is so, please adhere to the following instructions carefully:

`<Step 1>`

Zip up for re-submission files according to the following format :

<FT/PT>_Assn1_Resubmit_v1_<Stud. No.> _<Name>.zip

Example : **FT _ Assn1_Resubmit_v1_1234567_JohnDoeAnderson. zip**

- **<FT/PT>** Use "**FT**" for **F**ull-**T**ime student, "**PT**" if you are **P**art-**T**ime student

- Assn**1** if you are submitting assignment **1**, Assn**2** if submitting assignment **2** etc.

- **Resubmit_v1** if this is your **1**st re-submission

- **Resubmit_v2** if this is your **2**nd re-submission

- **<Stud. No.>** refers to your UOW student number (e.g. **1234567**)

- **<Name>** refers to your UOW registered name (e.g. **JohnDoeAnderson**)

- V. IMPT - To prevent Tutor's Inbox from blowing up in his face, each student is <u>only allowed to re-submit twice</u>, for each assignment only!

`<Step 2>`

Please email your single zip file to your tutor's email (refer to section 3) - Where to submit)

In your email **subject** line, type in the following information :

<FT/PT> <assignment info> <re-submission ver.> **<student number>** and **<name>**

Example:

To   :   **tutor's email** (refer to section 3) - Where to submit)

Subject   :   **FT** Assn**1** **Resubmit_v1** **1234567** **JohnDoeAnderson**