Lab 4

Due: Oct 3rd, 1:00PM.

Write your program in main.cpp and submit it.

This is a difficult lab. You can do as individual or a group. For group: (1) each group can have at most three students. (2) each group member submit individually; (3) Group leader should also submit a brief report. The report contains the group members' information and the work for each member.

Background:

This programming lab is designed to demonstrate the following areas of knowledge within C++:

- File access
- Multi-dimensional vectors
- Function design
- Flow control
- Decision making

The difficult part is intended to be the decision-making required to both parse the file input and to correctly draw the resulting vector.

Basics:

The first thing the program will do is ask the user for a file name to read in as input. The program will then read in this file, and parse the commands from it.

The program will, based on those commands, generate a two-dimensional vector of characters. The program will then output that vector to the console, and then repeat so long as there are more commands left in the input file.

Input Format:

Input to the program will consist of raw text in a series of words, numbers, or symbols. The commands used will be:

# rows – Generate the two-dimensional vector with the specified number of rows (note that "#" will be replaced with an integer, 1 - 9)

# columns – Generate the two-dimensional vector with the specified number of columns (note that “#” will be replaced with an integer 1- 9)

all c – Use the specified character (“c” will be replaced with a non-space, non-numeric character) to build the multi-dimensional vector. If this command is not issued in the sequence for one output, assume the “*” character.

triangular – build the two-dimensional vector in a triangular style; each row will have a number of columns equal to its ordinal position in the vector (note! The first row should have one column, not zero!)

outer – Any interior character in the multi-dimensional vector (i.e., one not on an edge of the rectangle or triangle “drawn” by outputting the characters) is a space. Note – this part is likely to be the most difficult to implement logically.

alphabetical – Build the vectors with the first drawn character “A”, and go down the alphabet from there.

go – Using the instructions given, build the multi-dimensional vector and write it out to the screen.

Some notes:

- The default character output is “*” if the “all” command is not entered.
- The file can have multiple command sets; one input file can be used to draw several vectors.
- Each time “go” is processed (and the vector built and displayed), reset all of the values to their defaults.
- “triangular” makes the column count unused, as does “alphabetical” make the “all” command unused
- Any command can come in any order. The two-part commands – rows, columns, all – will always immediately precede or trail their argument, though.
- No error processing will be needed for this assignment; you can assume that all input will be exactly as specified above, and not be missing any information to build a vector (e.g., all command sets will include a “# rows” command, and either “triangular” or “# columns”)

Output Format:

The output should just be the characters in the multi-dimensional vector, as defined above. For clarity and readability, though:

- Output each row with a space between each character output.
- Place a blank line between each vector output to the console.

Requirements:

In addition to the program functioning as presented, you must include:

- One function which builds the multi-dimensional vector
- A separate function which outputs the multi-dimensional vector to the screen.

Requirements:

Some functions you may use in this lab, do some research!!!!

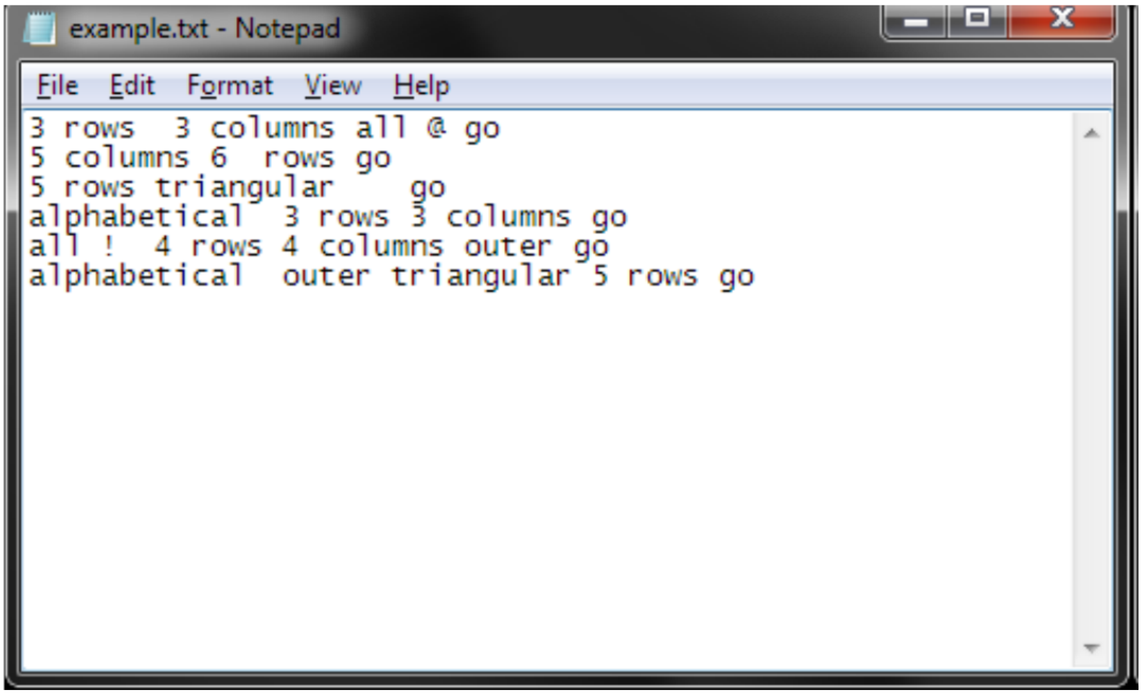
```
Ifstream::unget();  isspace(); isdigit();
vector::push_back();  vector::resize();

In addition, you should consider following

 vector<vector<char>> a;
Understand what is the return for size() function from
a.size(); what is the return for size() function from
a[0].size();
```

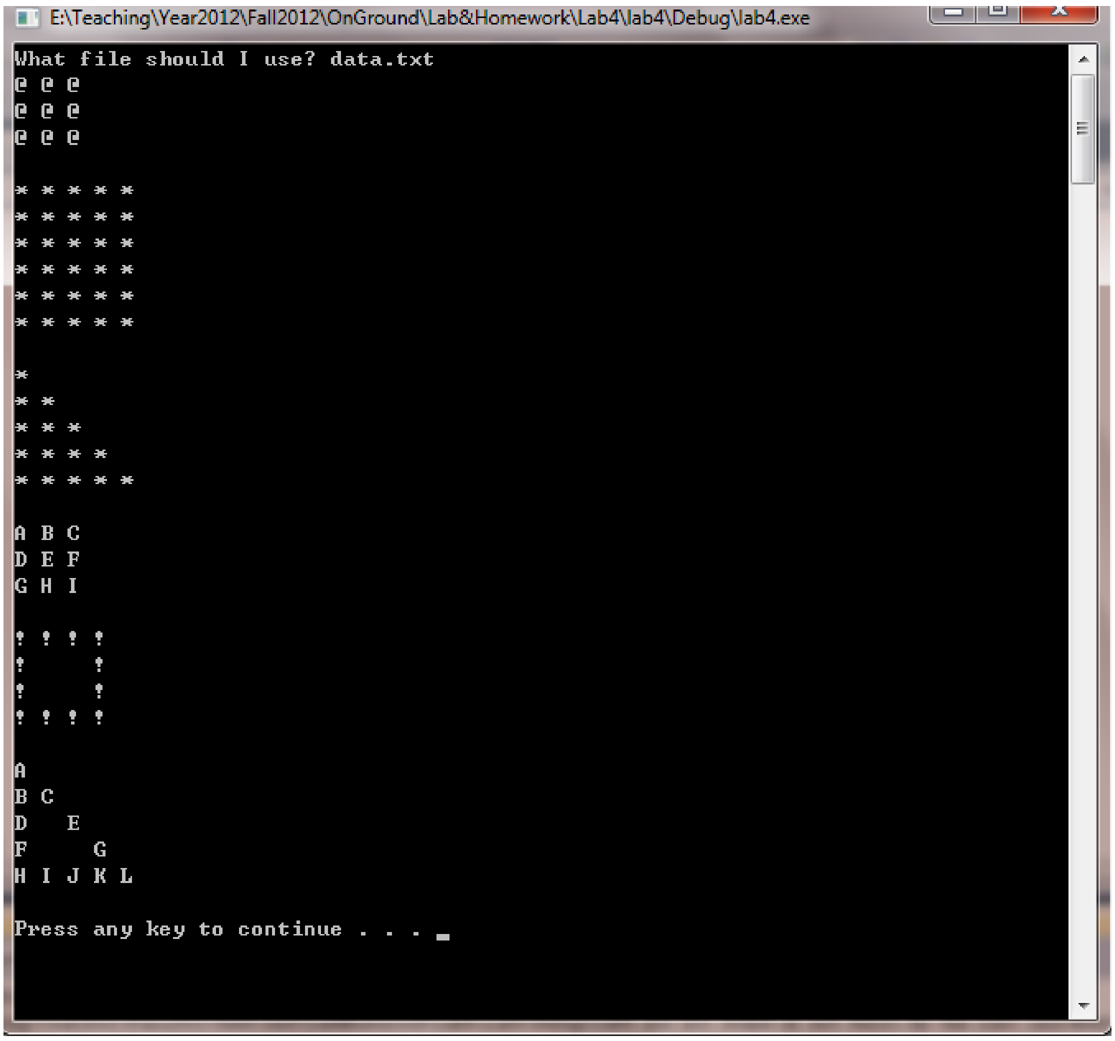
Examples:

Given the following input file:



```
3 rows  3 columns all @ go
5 columns 6  rows go
5 rows triangular    go
alphabetical  3 rows 3 columns go
all !  4 rows 4 columns outer go
alphabetical  outer triangular 5 rows go
```

The output would be:


E:\Teaching\Year2012\Fall2012\OnGround\Lab&Homework\Lab4\lab4\Debug\lab4.exe

```
What file should I use? data.txt
@ @ @
@ @ @
@ @ @

* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *

*
* *
* * *
* * * *
* * * * *

A B C
D E F
G H I

! ! ! !
!     !
!     !
! ! ! !

A
B C
D   E
F     G
H I J K L

Press any key to continue . . .
```