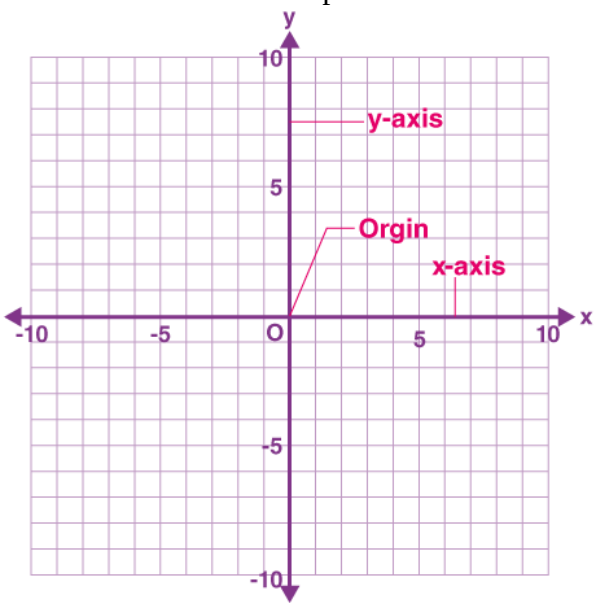
Lab4
Deadline: In lab in the week of Feb 12

This lab focuses on developing a program in C that reads a set of coordinates from an input file and uses them to draw lines by connecting the points. The coordinates are structured to form line segments, and the goal is to fill the space enclosed by these line segments with asterisks (*) in an output file. The figure shown below is a Cartesian plane.



Requirements
Write a function that draws a line following the coordinates and fill in the space. For this lab, there is no skeleton file provided, so you have to write your main function that interacts with the files (descriptions to follow).

You are given an input file which contains a set of (x, y) coordinates. These coordinates are written, so that a line segment is connected from the coordinate in the preceding line to the coordinate. For example, let's assume that input.txt contains the following coordinates:

>> cat input.txt
0,0
0,1
1,1
1,0
0,0
E
The character 'E' specifies the end of file.

The first coordinate is at coordinate 0,0 and the next coordinate is at 0,1. There is a line segment of a length 1 from 0,0 to 0,1 in the xy plane. The next line segment moves from 0,1 to 1,1 with a length of 1. This continues until line #5 is processed. In your output file (descriptions to follow),

the line segments are represented by a sequence of asterisks (*). From the previous input file, the output file will contain the following:

```
>>cat output.txt
**
**
```

Let's take a look at another example where we are drawing a rectangle this time.

```
>> cat input.txt
0,0
0,4
2,4
2,0
0,0
E
>> cat output.txt
***
***
***
***
***
```

In this case, the area enclosed by the line segment is filled with asterisks (*). Finally, your input and output can be the following as well.

```
>> cat input.txt
2,2
4,0
0,0
2,2
E
>> cat output.txt
   *
  ***
*****
```

Our program will read an input from a file and write an output to a file. There will not be any output to the console. In order to read/write from/to a file, we need to use file I/O libraries. I am not restricting on what functions you are using as long as they are a part of a standard library mentioned in learning hub. Here (https://www.tutorialspoint.com/cprogramming/c_file_io.htm) is a reference that might be useful to enable this.

Detailed specifications are below.

1. The command line arguments are in this order <executable> input_file output_file
2. You are guaranteed that your line segment will be strictly vertical, horizontal, or at 45 degree angle. In our example, the triangle was at 45 degrees.
3. You will write a main function as well as other helper functions.

4. Your input is in X,Y format and order. No spaces allowed between , X and Y coordinate.
5. You can assume that the coordinates can be any integer values. However, the test case will not request large memory where malloc fails due to insufficient free memory in the system.
6. There will not be line segment crossings.
7. The line segment will always come back to the original starting position.
8. There will be error case testing. If error case, write letter “Error” without double quotes to the output file.
9. The program should handle the error cases to write Error to output file and code must not break on error or invalid data.

Restrictions

- You must use malloc or a variant of malloc to allocate necessary memory. You are not allowed to allocate arbitrarily large array. In our triangle example, the malloc is supposed to only allocate 15B. Your malloc must be the smallest to be able to draw the shape. Your draw does not need to start from 0,0. Any malloc that uses extra space will get automatic 50% penalty in the overall grade.
- Only use standard libraries listed in Learning Hub.

How to Compile and Run

- The Makefile for lab4 is provided.
- The Makefile is supposed to work with lab4.c, input.txt, output.txt and ref.txt files so, make sure to name your files accordingly.
- Run the following command in vs code Terminal.

  `make`

  It should compile the code without any errors.

  `make convert_input`

  It should convert the input.txt file to unix encoding.

  `make run`

  It should run the compiled code.
- Run the following command to delete the out file.

  `make clean`
- Run the following command to convert the generated output to unix encoding.

  `make convert_output`

  It should convert the output.txt file to unix encoding.
- Run the following command to check your output with provided ref file.

  `make check`
- You are not supposed to make any changes in the Makefile.
- Make sure to install dos2unix utility using the following command:

  `sudo apt-get install dos2unix`

  For Mac

  `brew install dos2unix`

Grading

Any code that does not compile or breaks will result 0. For full marks this week, you must:

- (1 point) Correctly submitting the files with correct A numbers included.
- (6 point) Generate a correct solution to the problem(s) in this lab

Submission Files

- You must deliver only one .c file named: **lab4.c**
- The file that you submit should be a .c file (not .txt, not .cpp or any other type)
- Submit the file to learning hub before deadline.