Lab5
Deadline: In lab in the week of Feb 19

Requirements
lab5.c skeleton file is provided for you. Write a function zooms a 2D array of integers.

The signature of the function is:

```
void zoomArray(char **arr, float n, int *rows, int *cols);
```

You need to implement a function that zooms this array by an zoom factor specified int the input text file. The input array size is limited to the maximum of 30x30. You could get an input that is smaller than 30x30.
`int n` in `zoomArray` function is the expanding/shrinking factor. Below will give you an example with 2 as `n` value in `zoomArray`.

The input file will contain the following format:

First line: row value
Second line: column value
Third line: zoom factor
Next lines: Elements of the array
Last LIne: “E”to show End OF File

Here is a 2x2 example of the input:

```
2
2
2
11
22
E
```

Here is the output of running the input file through your program with a row size of 2,column size of 2 and zoom factor of 2. The original array orientation is retained, yet the whole image is enlarged by a factor of 2. Make sure that enlarging the image originates from the center of the image as in the example below.

```
1111
1111
2222
2222
```

Let’s assume another example of 4x4 example with a zoom factor of 0.5. Here is the input:

```
4
4
0.5
1111
1111
4444
4444
E
```

The output will be:

```
11
44
```

You might wonder if an input such as below will be given with a zoom factor of 0.5:

```
4
4
0.5
1111
2222
3333
4444
E
```

The answer is no since there is no way to make this shrink to 2x2. Since you will be guaranteed such conditions, this should make the algorithm simpler.

You will be guaranteed to have an input that is in squares. Any character will be used in the array in this lab. Also, the zoom factor will be guaranteed to be a number that can make the image shrink or expand.

Lastly, note that the function parameters are different in this lab. We are using array pointers instead of using array with []. Please review the different and understand what the differences are. This is main reason why the output print function has moved to the main function unlike the previous lab.

Requirement

- The input elements of array will be a value between 0-9.

How to Compile and Run

- The Makefile for lab5 is provided.
- The Makefile is supposed to work with lab5.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`

Restrictions

- You are not allowed to write any additional `printf` statement anywhere in the file.
- You are not allowed to modify any part of the code except rotateArray function, zoomArray function, and `a_num` variable.
- If you have any doubt, ask me during the lab session.

Grading

Any grading failure due to not following instructions will result in 0.

- (1 point) All files are submitted correctly using the instructions below.
- (3 point) Generate a correct solution to the problem(s) in this lab. Three test inputs will be used.

Submission Files

- You must submit only one .c file named: **lab5.c** (case sensitive) to learning hub**.**
- Make sure to update your A number. Look at the top of the file and write your A number including leading 0's.