Overview:

In this project, you will write a C program that utilizes command line arguments, file reading, file writing, and iterating through two dimensional arrays.

Restrictions:

You may only import `stdio.h`, `stdlib.h` and `string.h`

You may not use any global variables.

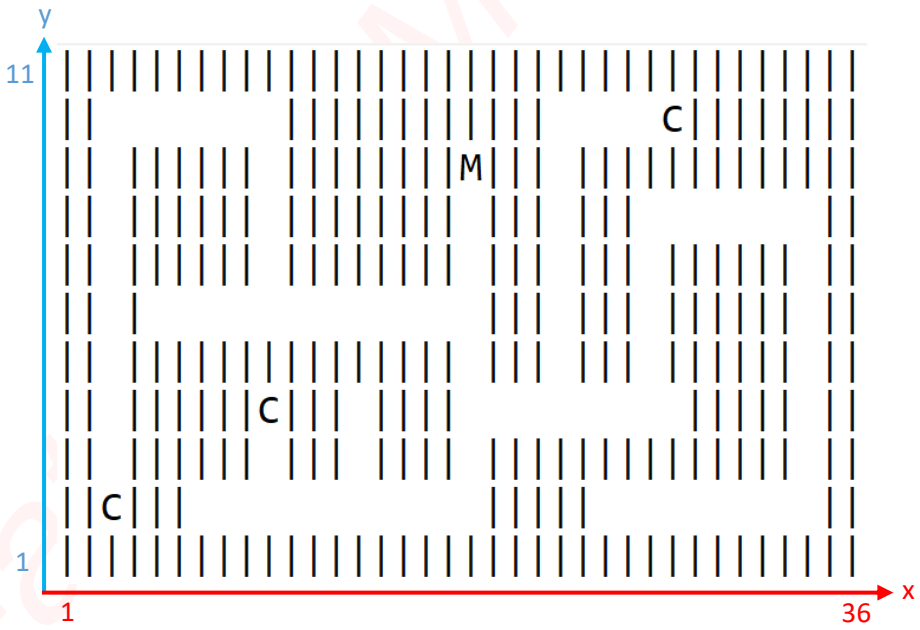
Premise:

Jerry the mouse has been captured by the evil scientist named Dr. Mays who has put him into a labyrinth. Jerry does not know which way to go within the labyrinth and is now hungry as he has stayed in place for some time. He smells some food nearby in the maze but is wary of wandering about endlessly. Your task is to help Jerry find the food within the maze so that he can satisfy his hunger.

Overview:

The maze has a grid layout with m x n rows and columns with the boundaries of the maze being indicated by the '|' character. Within the maze, there are one or more locations of food being indicated by the 'C' character. Also, Jerry's location is indicated at a random location by the 'M' character with the whitespace character ' ' indicating positions within the maze that Jerry can traverse.

Consider the maze below:



The bottom left boundary of the maze represents (x, y) coordinate (1,1) and the top right boundary of the maze represents (x, y) coordinate (36, 11)

Jerry's location within the maze is indicated by 'M' and his (x, y) coordinate is (19, 9)

Multiple cheeses (foods) are placed within the maze as indicated by 'C'. The (x, y) cheese coordinates are (3, 2), (10,4), and (28, 10)

Instructions:

Your source file will be named `p2_<userid>_<labsection>.c`

Your program will utilize **command line arguments** where two file names will be provided on the command line as follows:

```
./p2_<userid>_<labsection> input_file1.txt input_file2.txt
```

Input files:

The first input file will be the maze that you will utilize in your program. This maze is used by both the iterative search and the recursive search functions (see functions section for more information)

The second input file will contain multiple lines that contain one of the four cardinal directions (N, E, S, W) that indicate directions that Jerry will take within the maze during his iterative search but not during his recursive search.

Assumptions:

There will be at least one cheese within the maze
The input files will contain a single dot (.) and its extension afterwards (txt, in, etc.)
The length of a column for any particular maze will not exceed 50 characters
Each maze will contain a random location in which Jerry is located
Boundaries of the maze are indicated by '|'
Coordinates which can be traversed are indicated by ' ' (white space)
Mazes will contain at least a single path to a cheese (both iteratively and recursively)
Cardinal directions in the second input file will be each on their own line and will be a single valid capital letter character representing N, E, S, W
Mazes are fully enclosed, so Jerry cannot go out of bounds of any given maze
Rows of a given maze will have the same length

Output:

Your program will produce two separate output files where the first output files name will start with "iterative_" and then you will concatenate the file name the first input files name but with the extension .out. Within the first output file, it will contain the final maze where the maze indicates the paths taken by Jerry, Jerry's final location, and the (x, y) directions that were taken by Jerry in an iterative manner.

The second output files name will start with “recursive_” and then you will concatenate the file name the first input files name but with the extension .out. Within the second output, it will contain the final maze where the maze indicates the paths taken by Jerry, Jerry’s final location and the (x, y) directions that were taken by Jerry that only led to a cheese in a recursive manner.

Specific Requirements:

Your program must verify that there are two files input via command line arguments. If this condition is not held, then your program must print an error message and exit the program.

Your program must verify that the two input files can be opened. If this condition is not held, then your program must print an error message and exit the program.

Required functions (in addition to main):

You are free to choose any additional parameters for both of the functions below

```
void iterative_search(char ** maze, …)
```

The function utilizes the second input file from the command line arguments as directions that Jerry takes as he iteratively traverses through the maze.

This function keeps track of **every** (x, y) coordinate that Jerry has visited on his search for the cheese. Indicate coordinates that have been visited by the '.' character.

Once Jerry finds a cheese or if there are no more directions that Jerry can take, this function updates his final location in the maze.

```
int recursive_search(char ** maze, …)
```

This function does not utilize the second input file from the command line arguments.

For this function, your output may differ based off how you recurse through the maze – this is fine as long as you find at least a cheese.

This function returns 1 if the recursive search led to a cheese otherwise it returns 0 to indicate that the path that Jerry was on did not lead to a cheese.

This function keeps track of the (x, y) coordinates that **only** led Jerry to a cheese.

This function Indicates coordinates that have been visited by the '.' character.

Once Jerry finds a cheese, this function updates his final location in the maze.

Tips for recursively searching:

Consider the provided incomplete pseudocode as a starting point for your function:

```
FUNCTION  RecursiveSearch (parameters: char
pointer to pointer, …)
      IF current position contains food
            return successful search
      ENDIF
      IF current position has been visited prior
OR …
            return not a successful search
      ENDIF
      FOR …
            …
      ENDFOR
ENDFUNCTION
```

Makefile:

Create a Makefile similar to that of project 1 to compile your program.

Submission:

1. On zeus, create a directory named p2_<username>_<labsection>
   a. Copy your source file and Makefile to this directory
2. Create a typescript with the following content:
   a. Show that you are on zeus
   b. Show a listing of your directory
   c. Show your source code
   d. Remove any executable version using the Makefile
   e. Compile the code using the Makefile
   f. Run your program using the input files via command line arguments provided on Blackboard
   g. Show the contents of both output files
   h. Remove both output files
3. Be sure your directory ONLY contains the source file, typescript and Makefile
4. Change to the parent directory and create a tarfile of your project directory.
   a. Name this tarfile p2_<username>_<labsection>.tar
5. Submit this tarfile to Blackboard no later than the due date.

Sample Runs:

```
[hmughal2@zeus-2 p2]$ ./p2
Error - two input files were not provided as command line arguments
[hmughal2@zeus-2 p2]$ ./p2 labryinth.txt
Error - two input files were not provided as command line arguments
[hmughal2@zeus-2 p2]$ ./p2 labrynth.txt directions.txt
Error – the labrynth.txt file could not be opened
[hmughal2@zeus-2 p2]$ ./p2 labryinth.txt abc.txt
Error – the abc.txt file could not be opened
[hmughal2@zeus-2 p2]$ ./p2 labryinth.txt directions.txt
Success – two output files iterative_labryinth.out and
recursive_labryinth.out produced
```

iterative_labryinth.out and recursive_labryinth.out (see Blackboard for output files)

Both .out files contain the output maze as traversed by Jerry iteratively and recursively respectively with coordinates traversed being indicated by '.'. Both output mazes indicate Jerrys final coordinate.

Additionally, after the final output mazes there are directions taken by Jerry that indicate the coordinates taken where coordinates are in the (x, y) format.

In the instance of the iterative search, the output contains every direction (after the final output maze) that Jerry took in the cardinal direction with the x and y coordinate.

In the instance of the recursive search, the output contains after the final output maze) **ONLY** the cardinal directions that led Jerry to a cheese location **(NOT every direction that Jerry took)**.