CS241 – Data Organization
Fall 2022

Programming Assignment #7:
Image Processing with 2D arrays
Due by 11:59 PM on Friday, November 11, 2022

In this assignment, you will write a C program that involves processing two-dimensional arrays. A two-dimensional array is often used to represent a picture (or an image). For simplicity, your program will process only black-and-white images. Each pixel in the image will be a single char. The only legal chars are the asterisk ('*') which represents the color black, and the blank space (' ') which represents the color white. Your program will only be required to process **square** images, that is, the number of rows and columns in the image will be equal.

1. Assignment

Your program must read the image from **stdin**. The format of the input file is as follows: the first line of the file will contain a single integer followed immediately by a newline. This number is equal to the number of rows and columns in the image. Each succeeding line will contain one row of the image, followed by a newline char. For example, the input might be:

```
5
* * * * *
* *   * *
*   *   *
* *   * *
* * * * *
```

The size of the image will be at least 1x1. If the input file does not have this format, then your program should abort and return an exit status of 1. Your C program can abort processing and return to the command line by executing the statement:

```
exit(1);
```

Your program will need to #include <stdlib.h> in order to access the exit function.

Your program must implement the following image transformations.

- **invert:** This transformation should change every '*' into a blank character, and every blank character into a '*'

- **flip direction:** This transformation should flip the image. If the direction is either 'V' or 'v', then the image should be flipped across an imaginary vertical line down the center of the image. If the direction is either 'H' or 'h', then the image should be flipped across an imaginary horizontal line across the center of the image.

- **removeRedEye:** This transformation should change every '*' character into a blank character **if-and-only-if** the '*' does not have any other '*' adjacent to it, that is, all the adjacent locations are blanks. In other words, we want to "airbrush" away any isolated '*' characters in the picture. You should consider all **eight** adjacent cells.

The input image to your program should be read from the standard input (stdin), but the transformations to the image will be specified on the command line. You will invoke the program as follows:

```
$   ./a.out    list-of-desired-transformations    <   inputFileName
```

The syntax for the list-of-desired-transformations is as follows:

- **flip** – indicates a flip, and must be followed by a single char, which must be V, v, H, or h
- **inv** – indicates an inversion
- **redeye** – indicates that red eye should be removed

For example, the program might be invoked as:

```
$  ./a.out   flip   H   inv   redeye   flip V   <   inputFileName
```

Your program should perform each of the given transformations, in the given order, and then output the resulting image to **stdout**. Do **not** output the image size to stdout. Do **not** output any intermediate results to stdout. Do **not** output any other information, such as a message saying "Your image now looks like:"

If your program successfully transforms the image, then it should return an exit status of 0. If the command line arguments are in any way faulty, such as the spelling of the command is incorrect, or the flip is not followed by a legal direction, then your program should abort and return an exit status of 2. Notice that it is legal for the command line to contain zero transformations. In this case your program should simply output the original image.

I have provided some sample input files. You can copy the files provided for this assignment using the following command (do not forget to include the dot symbol at the end of this command):

```
$ cp   /nfs/faculty/soraya/cs241_fall2022/PA7/images/*  .
```

We strongly suggest that you read this statement in its entirety. Before writing a single line of code, not even the include or the main signature, you must understand the purpose and task to be performed by each transformation and design the algorithm that you will use to implement each of them. Understanding the problem clearly and even running a simple example by hand, will save you time at debugging.

2. What to submit

You should be sure to include **THE NAME OF THE AUTHOR OF THE PROGRAM** in a comment at the top of your source code file, for this and all other assignments in this course.

Your source code must use proper **style**, that is variables should be well named (name is not too short, not too long, and is meaningful), and bodies of loops, if's, etc.. should be properly indented. Refer to the coding style file for this class, published on Canvas under Coding Standards in a file named: cs241_codingStandards2020.pdf.

Create a .c file for this assignment and name it using your last name and the initial of your first name, like this: **lastName_initialFirstName_imageT.c** Submit this file for grading to Canvas in the place of this assignment.

3. General rubric

- +10 pts: Your C file follows the class coding style.
- +10 pts: Your submitted program provides an implementation for the **flip** function.
- +10 pts: Your submitted program provides an implementation for the **inv** function.
- +10 pts: Your submitted program provides an implementation for the **redeye** function.
- +15 pts: Your main function properly reads and validates input images and the transformations provided in the command line.
- +30 pts: Your object code passes a diff test with the image files provided in the assignment.
- +15 pts: Your object code passes a diff test with input files only available to the grader.