# C Programming II
# 2024 Fall
# Midterm

Instructor: Po-Wen Chi

Date: 2024.04.13 PM 2:00-6:00

**Policies**:

- Online test.
- Do not forget to include your Makefile. TA will only use the command make to build your program. If make fails, you will get zero points and no room for bargaining. **So if you do not know how to solve a problem, please, do not include it in your Makefile**.
- I do not care your source code file names, but the executive binary names should be **mid01, mid02, mid03, mid04, mid05**.
- You can ask TA if you do not understand the problems.

## 1 Matrix Multiplication (25 pts)

Please develop a program to calculate matrix multiplication. The user can input man matrices and your program needs to derive the product from them. The end of input will be a string "end". The usage is as follows.

```
$ ./mid01
1 matrix: [[1,0,0],[0,1,0],[0,0,1]]
2 matrix: [[2,2,2,2],[3,3,3,3],[4,4,4,4]]
3 matrix: [[1],[0],[0],[0]]
4 matrix: end
Result: [[2],[3],[4]]
```

The above example is for the following equation.

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 \\ 4 & 4 & 4 & 4 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 2 \\ 3 \\ 4 \end{bmatrix}$$

If there is any wrong input format, you should print **"Invalid Input"** and terminate the program. If the input matrices cannot be multiplied, you should print **"Multiplication Fails"** and terminate the program. For your simplicity, I promise all values are in **int32_t**.

## 2 Text Compression (25 pts)

In this class, I have shown you how a computer stores English symbols, like alphabets. Each character costs one byte memory. Unfortunately, this is not an efficient way to store characters. As you know, the frequency of each character is not equal. For example, the frequency of **"e"** is much higher than **"z"**. Therefore, if we use small memory size for **"e"** and large memory size for **"z"**, we can undoubtedly save memory space. Actually, the is a basic compression technique.

This time, I want you to develop a text compression program. Our TA will prepare a code book for you and your program needs to encode a ASCII text file to an output file. The code book is a text file and the following is a example.

```
space:
comma:
period:
a: 1111       // <-- 4 bits
b: 100000     // <-- 6 bits
...
e: 110        // <-- 3 bits
...
```

According to the above code book, **abe** will be encoded as 1111100000110. As you know, in computer memory, **byte** is the minimum unit instead of bit. So there will be a padding issue. The padding method is as follows. Append $10\ldots0$ until the the last byte is fulfilled. If the encoded data is a whole number of bytes long, you just need to append one additional byte 10000000 to the encoded data. So the final encoded result for **abe** is

$$1111100000110100 = \text{F8 34}.$$

As you can see, we need only two bytes to encode three characters. This time, I want you to develop an encoding program. The output file should be a binary file.

```
$ ./mid02
Codebook: codebook.txt
Input File: test.txt
Output File: output
$ hd output
00000000 F8 34
```

codebook.txt is given by the user. For your simplicity, I promise that the text file contains only lowercase alphabets and three symbols: space, comma, period. For any invalid choice, just print an error message and terminate the program. The output should be a binary file. I will provide some examples on the course site.

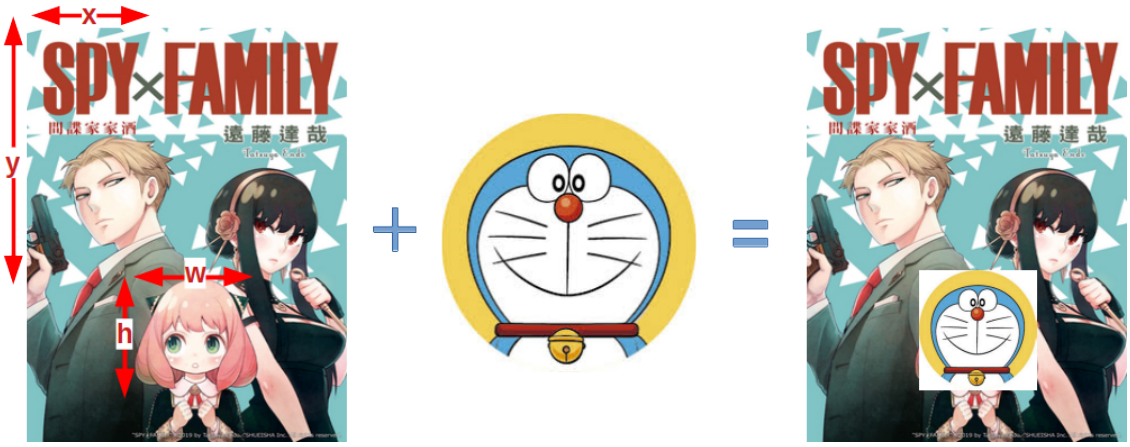



Figure 1: Example

# 3 Face/Off (25 pts)

I want you to develop a program to replace the face. Figure 1 is an example.

The usage is as follows.

```
$ ./mid03
cover: spy.bmp
x (in pixel): 300
y (in pixel): 600
w (in pixel): 50
h (in pixel): 50
new: doraemon.bmp
output: output.bmp
```

Note that $w \times h$ may be different to the embedded image. You need to resize it linearly. For any wrong input, simply print an error message and terminate the program.

# 4 Colorful Fantastic Render System [] (25 pts)

Look at Fig. 2. This is an image generated from the user command, which you can see the command at the bottom of Fig. 2. TA Howard loves digital art, especially when it's made with just a few simple commands. One day, DarthMaul shared a cool tool, CFRS[], which can generate Fig. 2 by using only C, F, R, S, [ and ]. In this problem, you need to create a Colorful Fantastic Render System, i.e., CFRS[] for TA Howard.

## 4.1 Commands

Most commands are similar to CFRS[]. Do not worry! I will make it easier. I will show you the command definition used in this problem here:

- C: Change colour to the next one from the cyclic sequence:
  black (#000000) → blue (#3366ff) → green (#00cc00) → cyan (#00cccc), red (#cc0000) → magenta (#cc00cc) → yellow (#cccc00) → white (#ffffff) → black.

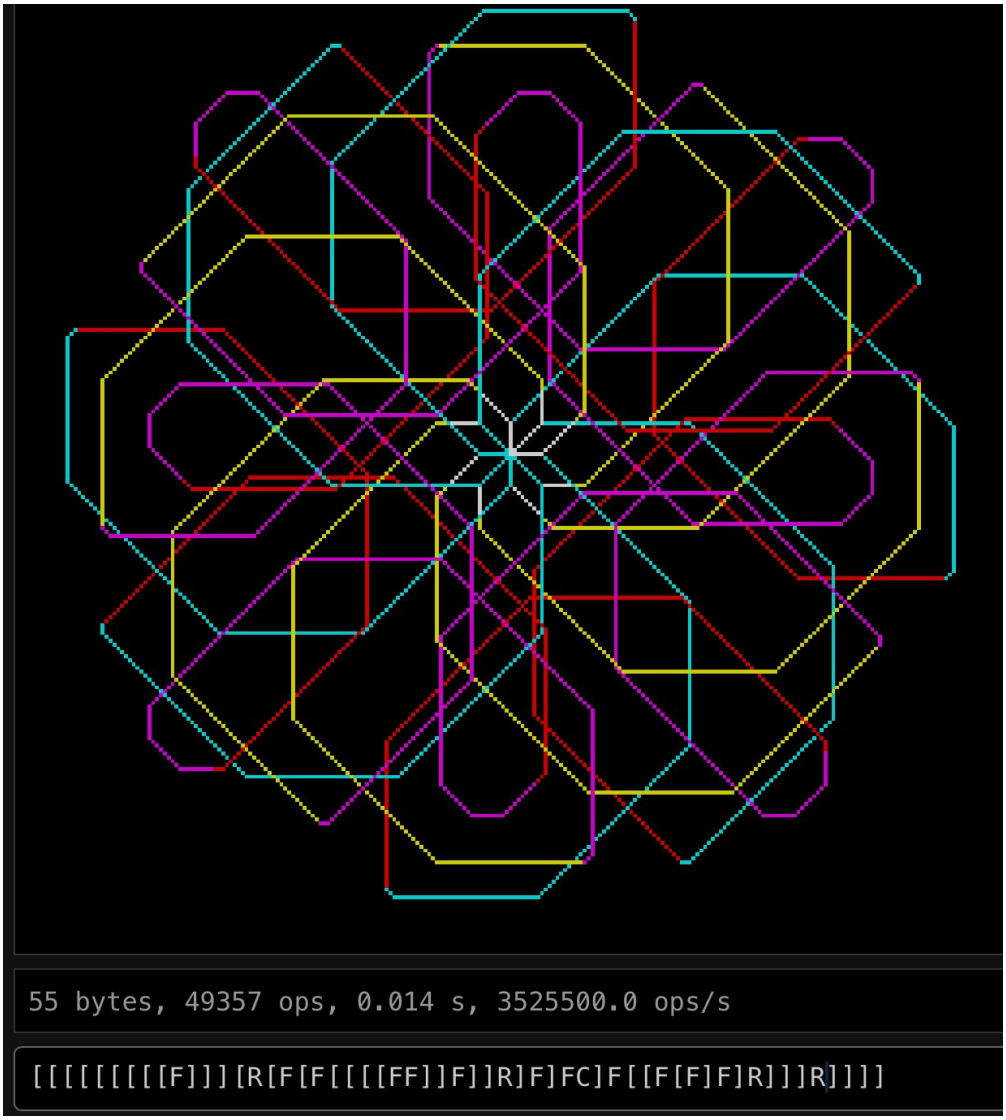



Figure 2: The amazing art from commands.

- **F**: Move forward by one pixel and paint the new pixel.
- **R**: Rotate clockwise by 45°.
- []: Indicates a loop structure with [ marking the start of a repeatable block and ] signifying the end. The code within these brackets executes **twice** before exiting the loop. For example, [F]=FF, [[F]]=FFFF.

## 4.2 Setting

- Image size: 256 × 256 pixel.
- The initial location is at the center (127,127).

- The starting orientation (0°) is North (up).
- The initial color is White.

## 4.3 User Interface

```
$ ./mid04
Enter the output filename: output.bmp
Enter the background color (R,G,B): (0,0,0)
Type code here: [commands]
```

You need to generate the BMP image according to the background color and the user code.

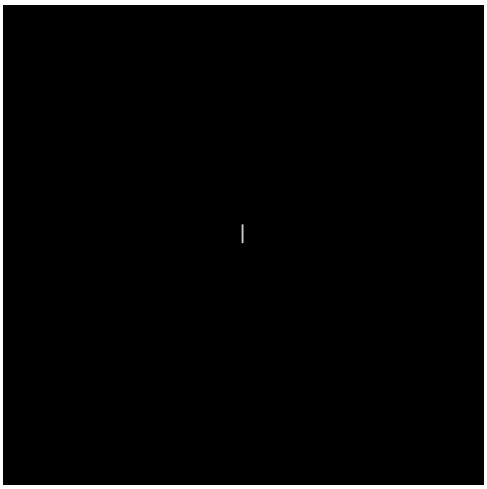
## 4.4 Examples

There are some examples here. All examples are with black backgrounds.

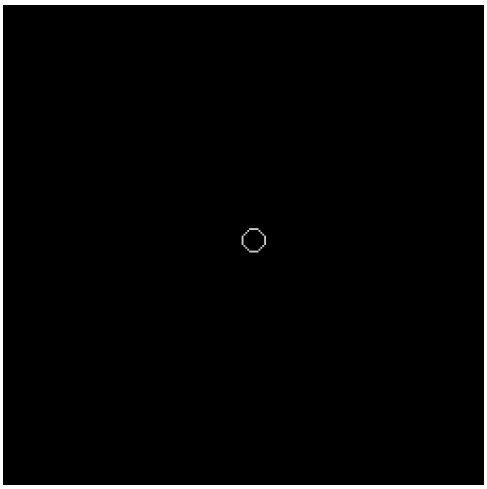
1. Code: FFFFFFFFFF
   - The output image is Fig. 3a
   - 10 F, so go up 10 pixel.
2. Code: FFFFRFFFFRFFFFRFFFFRFFFFRFFFFRFFFFRFFFFR
   - The output image is Fig. 3b
   - Go up 4 pixel, then turn right 45°, repeat it 8 times, so the final graph is a circle.
3. Code: FFFFCFFFFCFFFFCFFFFCFFFFCFFFFCFFFFCFFFFCFFFFCFFFFCFFFFC
   - The output image is Fig. 3c
   - Change a color every 4 pixels. The initial color is White, and the second color is Black.
4. Code: [[[[[[[[[F]]]]R]]RR]]RRCC]]
   - The output image is Fig. 3d
   - TA Jacob generate a pseudo code to explain this example, please see the appendix.

## 4.5 Grading

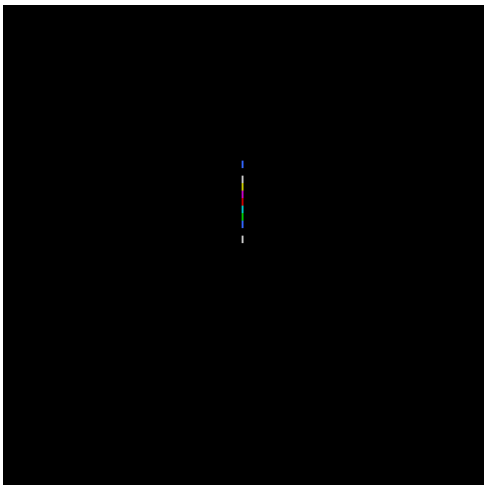
- Follow the input format and order (TA can grade without extra effort): 1 pt
- Image and background color configuration: 4 pts
- Contain only **F**: 4 pts



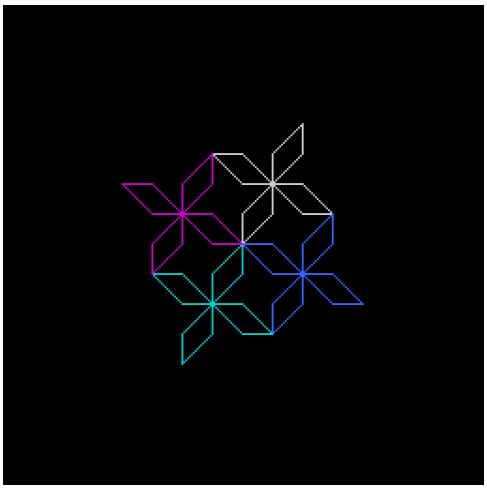
(a) Code: FFFFFFFFFF



(b) Code: FFFFRFFFFRFFFFRFFFFRFFF-FRFFFFRFFFFRFFFFR



(c) Code: FFFFCFFFFCFFFFCFFF-FCFFFFCFFFFCFFFFCFFFFCFFFFCFFF-FCFFFFC



(d) Code: [[[[[[[[[[F]]]]R]]RR]]RRCC]]

- Contain only **F,R**: 4 pts
- Contain only **F,C**: 4 pts
- Contain only **F,R,C**: 4 pts
- All **C,F,R,[,]**: 4 pts

For your simplicity, all inputs will be valid. That is, you do not need to handle error cases.

## 4.6 Hint

1. **scanf** is good enough to handle input (0,0,0).
2. The following array may help with the Rotation:

```
di[8]={-1,-1,0,1,1,1,0,-1};
dj[8]={0,1,1,1,0,-1,-1,-1};
```

3. I think recursive may help to solve [].
4. Good Luck!

## 4.7 Appendix

The pseudo code for example 04.

```
Begin loop:
    Begin loop:
        Begin loop:
            Begin loop:
                Begin loop:
                    Begin loop:
                        Begin loop:
                            Begin loop:
                                Begin loop:
                                    Begin loop:
                                        Move forward and paint the cell.
                                    End loop.
                                End loop.
                            End loop.
                        End loop.
                        Rotate direction by 1/8th of a turn.
                    End loop.
                End loop.
                Rotate direction by 1/8th of a turn.
                Rotate direction by 1/8th of a turn.
            End loop.
        End loop.
        Rotate direction by 1/8th of a turn.
        Rotate direction by 1/8th of a turn.
```

```
        Change color to the next in predefined list.
        Change color to the next in predefined list.
    End loop.
End loop.
```

# 5 Bonus: Your Comments (5 pts)

Your comments about this class. Any comments are welcomed. Do not worry about typos or any grammar errors. However, you will get nothing if you leave this question blank.