C Programming II
2020 Spring
Final

Instructor: Po-Wen Chi

Due: 2020.06.24 PM 15:30

Policies:

- Offline test unless you complete the exam and want to submit your code.
- 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 **fin01, fin02, fin03, fin04, fin05**.
- You can ask TA if you do not understand the problems.

1 BMP: Less Colors (25 pts)

Digital images consist out of pixels. If we want to store or transmit an image, we have to provide information about each individual pixel. A common representation of color information is to use 3 bytes to represent RED, GREEN, BLUE respectively. This is what I have shown you in this class. However, when I was young, we did not use 3 bytes to present a pixel. Instead, we preferred to use **2 bytes** to present one pixel. So there are only 65535 possible colors. How to represent 3 colors in only 2 bytes? We use 5 bits for Red and Blue and 6 bits for Green. The transformation is presented in figure 1. Note that since we have less bits (information) available, we can represent less colors.

For your reference, I provide you BMP wikipedia and two BMP files, one is 3-bytes presentation and the other is 2-bytes presentation. Your program should be

```
$ ./fin01
Please enter the input image name: maldives.bmp
Please enter the output image name: maldives_16.bmp
Done!
```

Note that the user may input a file which is not a BMP file, you need to give a warning. How to check if the input is a BMP file? Read the first two bytes.

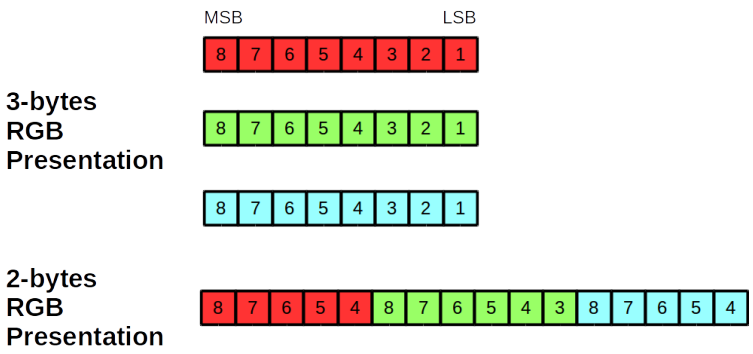

Figure 1: Color Transformation.

2 CPU Usage (25 pts)

I want you to monitor the CPU usage of your computer. Do not worry, I will tell you how to get the CPU information. You can use the following command.

```
cat /proc/stat
```

You do not need to care cpu0, cpu1, ...You just need to read the line starts with cpu. Do you know what those numbers mean? I will not tell you. Please read the manual.

```
man 5 proc
```

Give you a hint. Do you know how to search in the manual? The search keyword is **proc/stat**. What you need to do is to summation all cpu usage times, minus the cpu idle time and calculate the ratio.

```
$ ./fin02
CPU usage: 5.12%
CPU usage: 6.73%
...
```

Your program should run continuously until Ctrl+C.

3 Matrix Operation: Implementation by Linked-List (50 pts)

In this class, I have shown you how Linus developed a linked-list. Now I want you to use this to develop a matrix structure, as shown in figure 2. **Note that using two dimensional array is not allowed**.

Note that in your design, you should take the sparse matrix into account. That is, given a matrix $M_{100\times200}$ where $M[0][0] = 1, M[99][0] = 2, M[99][199] = 3$ and others are all zeros, your structure should look like figure 3. As you can see, this implementation saves lots of memory.

Please implement the following things:

```
typedef struct _sMatrix
```

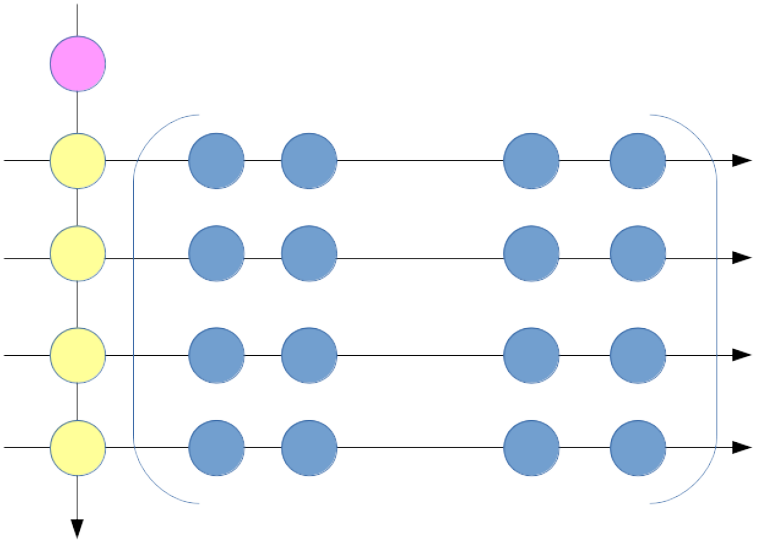

Figure 2: Construct a matrix using linked-list.

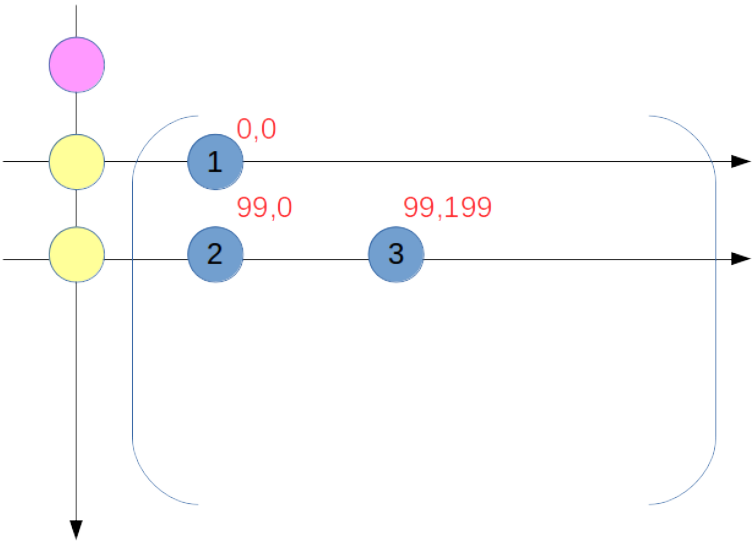

Figure 3: Sparse matrix.

```
{
} sMatrix;

void set_matrix( sMatrix *m, int a, int b, int val ); // m[a][b] = val
void free_matrix( sMatrix *m )
void print_matrix( sMatrix *m );
int add_matrix( sMatrix *output, sMatrix a, sMatrix b ); // output = a + b
                                                         // return -1 if fail
int mul_matrix( sMatrix *output, sMatrix a, sMatrix b ); // output = a * b
                                                         // return -1 if fail
void transpose_matrix( sMatrix *m );
int det_matrix( sMatrix *m );                            // Calculate the
   determinant.
```

The free function must free all allocated memories.

The print format is as from left to right and then top to down. So the figure 3 output is

```
(0,0,1)
(99,0,2)
(99,199,3)
```

If you do not know how to calculate the determinant, please check the wikipedia or google.

`https://zh.wikipedia.org/wiki/%E8%A1%8C%E5%88%97%E5%BC%8F`

TA will prepare test.c for you. You need to build test.c in your Makefile.

The points distribution is as follows:

- structure: 5 pts
- set: 5 pts
- free: 5 pts
- print: 5 pts
- add: 5 pts
- mul: 5 pts
- transpose: 10 pts
- determinant: 10 pts

4 Bonus: Your Comments (5 pts)

At last, you finish this class. Again, any comments are welcomed. However, you will get nothing if you leave this question blank.