lab14

```
$ gcc lab14.c
$ ./a.out ../pic1.ppm ../EE.ppm ../NTHU.ppm out.ppm
score: 70
o. [Output] Program output is correct, good.
o. [Format] Program format can be improved
o. [NTHU] logo should be water-marked.
o. [PPMcvt] needs to return a new Image, not modifying the old one.
```

lab14.c

```c
// EE231002 Lab14. Image Processing
// 109061158, 簡佳吟
// Date: 2021/1/4

#include <stdio.h>
#include <stdlib.h>

typedef struct sPIXEL {     // a single pixel
    unsigned char r, g, b;  // three color components

} PIXEL;

typedef struct sIMG {       // an image of PPM style
    char header[3];         // header, either P3 or P6
    int W, H;               // width and height of the image
    int level;              // intensity level of each color component
    PIXEL **PX;             // two-dimension array for all the pixels

} IMG;

IMG *PPMin(char *inFile);
    // This function opens the inFile, reads the image data
    // and returns a pointer pointing to the newly created
    // image and data structure
void PPMout(IMG *pic, char *outFile);
    // This function writes the image pointed by p1
    // to the output file outFile

IMG *PPMcvt(IMG *pic, IMG *ee, IMG *nthu, int x1, int y1, int x2, int y2);
    // This function processes the image pointed by p1
    // performing the modifications stated above and returns the new image
    // as a result. The argument x1, y1 are the coordinates of the
    // lower-left corner of the box which retains the color image;
    // while x2, y2 are the coordinates of the corner of the box.
int main(int argc, char *argv[])
{
    IMG *pic, *ee, *nthu;       // image

    pic = PPMin(argv[1]);           // read its data
    ee = PPMin(argv[2]);
```

```
41      nthu = PPMin(argv[3]);
42      PPMout(PPMcvt(pic, ee, nthu, 3500, 2000,3700, 1800), argv[4]); // output
        PPMout(PPMcvt(pic, ee, nthu, 3500, 2000, 3700, 1800), argv[4]); // output
43      return 0;    // done and return
44  }
45
46  // This function opens the inFile, reads the image data
47  // and returns a pointer pointing to the newly created
48  // image and data structure
49  IMG *PPMin(char *inFile)
50  {
51      int i, j;    // index for loop
52      IMG *pic;        // the image
53      FILE *fin;       // the file
54
55      pic = (IMG *)malloc(sizeof(IMG));
56      fin = fopen(inFile, "r");            // open file
57      fscanf(fin, "%s", pic->header);      // scanf the image
58      fscanf(fin, "%d %d\n%d\n", &pic->W, &pic->H, &pic->level);
59      pic->PX = (PIXEL **)malloc(pic->W * sizeof(PIXEL *));
60
61      for(i = 0; i < pic->W; i++) {
        for (i = 0; i < pic->W; i++) {
62          pic->PX[i] = (PIXEL *)malloc(pic->H * sizeof(PIXEL));
63      }
64      for (j = 0; j < pic->H; j++) {           // scan the color component
65          for (i = 0; i < pic->W; i++) {       // of each pixel
66              pic->PX[i][j].r = getc(fin);
67              pic->PX[i][j].g = getc(fin);
68              pic->PX[i][j].b = getc(fin);
69          }
70
71      }
72      fclose(fin);     // close the file
73      return pic;      // done and return
74  }
75
76  // This function processes the image pointed by p1
77  // performing the modifications stated above and returns the new image
78  // as a result. The argument x1, y1 are the coordinates of the
79  // lower-left corner of the box which retains the color image;
```

```
80 // while x2, y2 are the coordinates of the corner of the box.
81 IMG *PPMcvt(IMG *p1, IMG *ee, IMG *nthu, int x1, int y1, int x2, int y2)
82 {
83     int x, y, i, j;          // index for loop
84     unsigned char gray;      // the color component of gray color
85     PIXEL block = {0, 255, 255};    // cyan color block component
86
87     // set the pic to black-white
88     for (y = 0; y < p1->H; y++) {
89         for (x = 0; x < p1->W; x++) {
90             if (x < x1 || y < y2 || x > x2 || y > y1) {
91                 gray = p1->PX[x][y].r * 0.2126;
92                 gray += p1->PX[x][y].g * 0.7152;
93                 gray += p1->PX[x][y].b * 0.0722;
94                 p1->PX[x][y].r = gray;
95                 p1->PX[x][y].g = gray;
96                 p1->PX[x][y].b = gray;
97             }
98         }
99     }
100
101    // add block on me
102    // horizontal line
103    for (x = x1; x <= x2; x++) {
104            p1->PX[x][y1] = block;
       p1->PX[x][y1] = block;
105            p1->PX[x][y2] = block;
       p1->PX[x][y2] = block;
106    }
107    // vertical line
108    for (y = y2; y <= y1; y++) {
109            p1->PX[x1][y] = block;
       p1->PX[x1][y] = block;
110            p1->PX[x2][y] = block;
       p1->PX[x2][y] = block;
111    }
112    // paste EE logo
113    for (y = 0, j = p1->H - ee->H - 200; y < ee->H && j < p1->H; y++, j++) {
114        for (x = 0, i = (p1->W - ee->W) / 2 ; x < ee->W && i < p1->W ;
       for (x = 0, i = (p1->W - ee->W) / 2; x < ee->W && i < p1->W;
115                                                          x++, i++) {
```

```
116             // ignore the white color
117             if (ee->PX[x][y].r != 255 && ee->PX[x][y].g != 255
118                                         && ee->PX[x][y].b != 255) {
119
120                 p1->PX[i][j] = ee->PX[x][y];
121
122             }
123         }
124     }
125     // paste NTHU logo
126     for (y = 0, j = 0; y < p1->H && j < nthu->H; y++, j++) {
127         for (x = 0, i = 0; x < p1->W && i < nthu->W; x++, i++) {
128             // ignore the white color
129             if (nthu->PX[x][y].r != 255 && nthu->PX[x][y].g != 255
130                                             && nthu->PX[x][y].b != 255) {
131
132                 p1->PX[x][y].g = nthu->PX[i][j].g;
133                 p1->PX[x][y].r = p1->PX[x][y].b = 255;
134             }
135         }
136     }
137
138     return p1;      // done and return
139 }
140
141 // This function writes the image pointed by p1
142 // to the output file outFile
143 void PPMout(IMG *p1, char *outFile)
144 {
145     int i, j;       // index for loop
146     FILE *fout;     // the outputfile
147
148     fout = fopen(outFile, "w");     // open the image
149     fprintf(fout, "%s\n%d %d\n%d\n", p1->header, p1->W, p1->H, p1->level);
150         // print the data
151     for (j = 0; j < p1->H; j++) {       // print the color component
152         for (i = 0; i < p1->W; i++) {   // of each pixel
153             fprintf(fout, "%c%c%c", p1->PX[i][j].r, p1->PX[i][j].g,
154                                                     p1->PX[i][j].b);
155
156         }
```

```
157     }
158     fclose(fout);           // close the file
159 }
```