

(1) 다음 영상에 대한 영상 히스토그램을 그리는 프로그램을 작성하라

입력영상: /image/puppy.pgm (10점)

```
#include <stdio.h>
int main(int argc, char * argv[])    //argc는 외부에서 입력한 문자열 개수,
{                                     //argv[]는 외부에서 입력한 문자열들
    int image1[600][800];             //800x600 영상까지 처리가능
    int image2[256][256];             //256x256 히스토그램
    int image3[256][256];             //256x256 거꾸로 히스토그램

    int x, y;                         //반복문에서 사용할 정수 변수 x, y 선언
    char M, N;                       //문자 변수 M과 N 선언
    int XX, YY, MAX;                 //정수 변수 XX, YY, MAX 선언
    FILE *file1, *file2;             //파일 변수 file1과 file2 선언

    int frequency[256]={0};          //명암도의 빈도를 체크할 변수 선언후 초기화
    int bright, height;              //명암도 정수변수 bright와 높이 정수 변수 height 선언

    file1 = fopen(argv[1], "r");      //외부에서 입력한 두 번째 문자열을 읽기모드로 열
    fscanf(file1, "%c", &M);          //헤더의 매직넘버를 읽어 들임 'P'
    fscanf(file1, "%c", &N);          //헤더의 매직넘버를 읽어 들임 '2'
    fscanf(file1, "%d", &XX);         //입력영상의 가로크기 읽어 들임
    fscanf(file1, "%d", &YY);         //입력영상의 세로크기 읽어 들임
    fscanf(file1, "%d", &MAX);        //입력영상의 최대 명암도 읽어 들임

    for(y = 0; y < YY; y++)          //반복문을 사용하여 입력영상의 픽셀 값 읽어 들임
        for(x = 0; x < XX; x++)
            fscanf(file1, "%d", &image1[y][x]);

    for(y = 0; y < YY; y++)
        for(x = 0; x < XX; x++)
        {
            bright = image1[y][x];    //이미지의 픽셀 값을 bright에 넣음
            frequency[bright] = frequency[bright] + 1; //같은 픽셀 값이 나오면
        }                             //해당 픽셀 값의 빈도수 증가

    for(bright=0; bright<256; bright++)
    {
        frequency[bright] = frequency[bright] / 4;    //높이를 낮춰줌
        if(frequency[bright] > 256) frequency[bright] = 256; //255보다 높으면 255로 고정
        for(height=0; height < frequency[bright]; height++)
    }
```

```

    image2[height][bright] = 255;    //해당하는 픽셀 값에 빈도수 높이만큼 채워줌
}
XX=256;                            //히스토그램의 가로축 크기
YY=256;                            //히스토그램의 세로축 크기

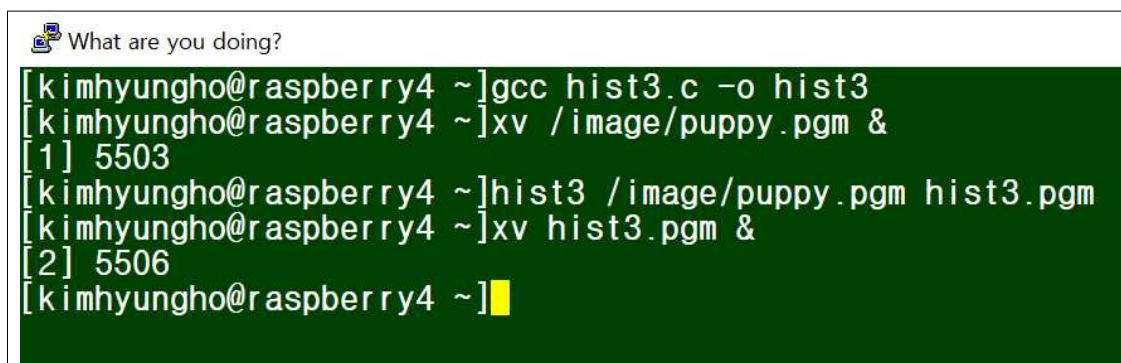
for(y = 0; y < YY; y++)
    for(x = 0; x < XX; x++)
        image3[(YY-1)-y][x] = image2[y][x]; //히스토그램을 X축 대칭 시킴

file2 = fopen(argv[2], "w"); //외부에서 입력한 세 번째 문자열을 쓰기모드로 엮
fprintf(file2, "%c", M);    //읽어 들인 두 번째 문자열 파일의 헤더 매직넘버를
fprintf(file2, "%cWn", N);  //세 번째 문자열 파일에 입력
fprintf(file2, "%d %dWn", XX, YY); //읽어 들인 가로축과 세로축 크기를 입력
fprintf(file2, "%dWn", MAX); //읽어 들인 최대 명암도를 입력

for(y = 0; y < YY; y++){
    for(x = 0; x < XX; x++){ //반복문을 이용하여 히스토그램을
        fprintf(file2,"%4d", image3[y][x]); //세 번째 문자열 파일에 입력
    }
    fprintf(file2, "Wn");    //줄 내림 입력
}

return 0;
}

```

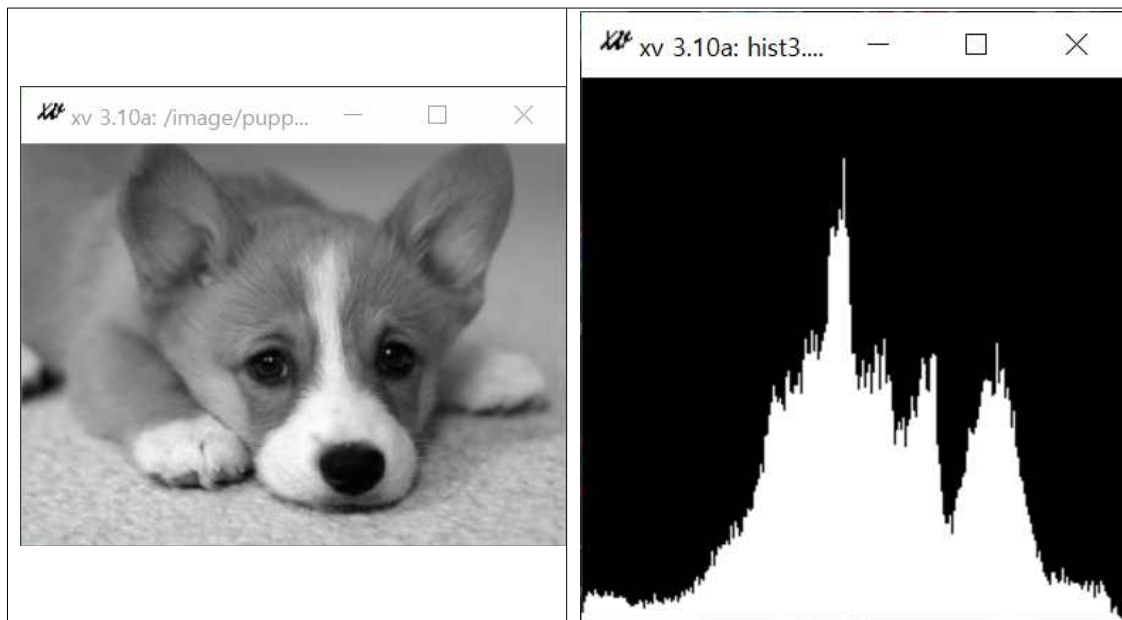


A terminal window titled "What are you doing?" shows the following commands and output on a dark green background with white text:

```

[kimhyungho@raspberrypi4 ~]gcc hist3.c -o hist3
[kimhyungho@raspberrypi4 ~]xv /image/puppy.pgm &
[1] 5503
[kimhyungho@raspberrypi4 ~]hist3 /image/puppy.pgm hist3.pgm
[kimhyungho@raspberrypi4 ~]xv hist3.pgm &
[2] 5506
[kimhyungho@raspberrypi4 ~]

```



fscanf로 읽기모드로 연 입력영상 /image/puppy.pgm의 헤더와 데이터 포맷을 읽어 들였습니다. 읽어 들인 데이터 포맷의 픽셀 값을 반복문을 통해서 한번 씩 bright에 넣고 frequency[bright] = frequency[bright] + 1을 통해서 같은 픽셀 값이 나올 때 마다 빈도수를 증가 시켰습니다. 그리고 같은 픽셀 값의 빈도수가 너무 많아서 세로축이 너무 커져 나누기 4를 해서 높이를 줄였습니다. 만약 높이를 줄였음에도 불구하고 세로축의 크기가 255가 넘어간다면 조건문을 통해 255로 해당 픽셀 값을 고정되게 했습니다. 그리고 반복문을 통해서 히스토그램의 가로축이 픽셀 값이고 세로축 높이가 빈도가 되게 만들었습니다. 그리고 fprintf를 이용해서 쓰기모드로 연 hist3.pgm에 읽어 들인 영상의 헤더 부분과 히스토그램의 데이터 포맷부분을 입력했습니다. 실행과정에서 hist3.c를 컴파일하고 실행파일을 hist3로 만든 뒤 hist3로 /image/puppy.pgm의 히스토그램인 hist3.pgm을 만들고 후면 작업을 통해 입력 영상인 /image/puppy.pgm과 히스토그램인 hist3.pgm을 동시에 출력했습니다.

(2)다음 컬러 영상에 대한 영상 히스토그램을 그리는 프로그램을 작성하라

입력영상: /image/puppy.ppm (30점)

```
#include <stdio.h>
#include <stdlib.h>          //입력받은 4번째 문자열을 정수화 시키기 위해
int main(int argc, char * argv[]) //argc는 외부에서 입력한 문자열 개수,
{
    //argv[]는 외부에서 입력한 문자열들
    int image1[600][800*3]; //800x600 영상까지 처리가능 컬러영상
    int image2[256][256*3]; //256x256 히스토그램 컬러영상
    int image3[256][256*3]; //256x256 거꾸로 히스토그램 컬러영상

    int x, y;                //반복문에서 사용할 정수 변수 x, y 선언
    char M, N;               //문자 변수 M과 N 선언
    int XX, YY, MAX;         //정수 변수 XX, YY, MAX 선언
    int n = atoi(argv[3]);   //입력받은 4번째 문자열을 정수화 시켜 n에 넣음
    FILE *file1, *file2;     //파일 변수 file1과 file2 선언

    int frequency[256]={0}; //명암도의 빈도를 체크할 변수 선언후 초기화
    int bright, height;      //명암도 정수변수 bright와 높이 정수 변수 height 선언

    file1 = fopen(argv[1], "r"); //외부에서 입력한 두 번째 문자열을 읽기모드로 열
    fscanf(file1, "%c", &M);     //헤더의 매직넘버를 읽어 들임 'P'
    fscanf(file1, "%c", &N);     //헤더의 매직넘버를 읽어 들임 '3'
    fscanf(file1, "%d", &XX);    //입력영상의 가로크기 읽어 들임
    fscanf(file1, "%d", &YY);    //입력영상의 세로크기 읽어 들임
    fscanf(file1, "%d", &MAX);   //입력영상의 최대 명암도 읽어 들임

    for(y = 0; y < YY; y++) //반복문을 사용하여 입력영상의 픽셀 값 읽어 들임
        for(x = 0; x < XX*3; x++) //컬러영상은 가로픽셀 당 값을 3개 가지므로 x3
            fscanf(file1, "%d", &image1[y][x]);

    for(y = 0; y < YY; y++)
        for(x = n; x < XX*3; x+=3) //n = 0 이면 R, n = 1 이면 G, n = 2이면 B
        {
            bright = image1[y][x]; //픽셀 값을 bright에 넣음
            frequency[bright] = frequency[bright] + 1; //같은 픽셀 값이 나오면
        } //해당 픽셀 값의 빈도수 증가

    for(bright=0; bright<256; bright++)
    {
        frequency[bright] = frequency[bright] / 4; //히스토그램의 높이를 낮춰줌
        if(frequency[bright]>256) frequency[bright] = 256; //255보다 높으면 255로 고정
    }
```

```

for(height=0; height < frequency[bright]; height++)
    image2[height][bright*3+n] = 255; //해당하는 픽셀 값에 빈도수 높이만큼 채워줌
    //R, G, B에 맞게 값을 분배

XX=256; //히스토그램의 가로 크기
YY=256; //히스토그램의 세로 크기

for(y = 0; y < YY; y++)
for(x = 0; x < XX*3; x++) //컬러영상은 가로픽셀 당 값을 3개 가지므로 x3
    image3[(YY-1)-y][x] = image2[y][x]; //히스토그램을 X축 대칭 시킴

file2 = fopen(argv[2], "w"); //외부에서 입력한 세 번째 문자열을 쓰기모드로 엮
fprintf(file2, "%c", M); //읽어 들인 두 번째 문자열 파일의 헤더 매직넘버를
fprintf(file2, "%cWn", N); //세 번째 문자열 파일에 입력
fprintf(file2, "%d %dWn", XX, YY); //읽어 들인 가로축과 세로축 크기를 입력
fprintf(file2, "%dWn", MAX); //읽어 들인 최대 명암도를 입력

for(y = 0; y < YY; y++){
    for(x = 0; x < XX*3; x++){ //반복문을 이용하여 컬러 히스토그램을
        fprintf(file2,"%4d", image3[y][x]); //세 번째 문자열 파일에 입력
    }
    fprintf(file2, "Wn"); //줄 내림 입력
}

return 0;
}

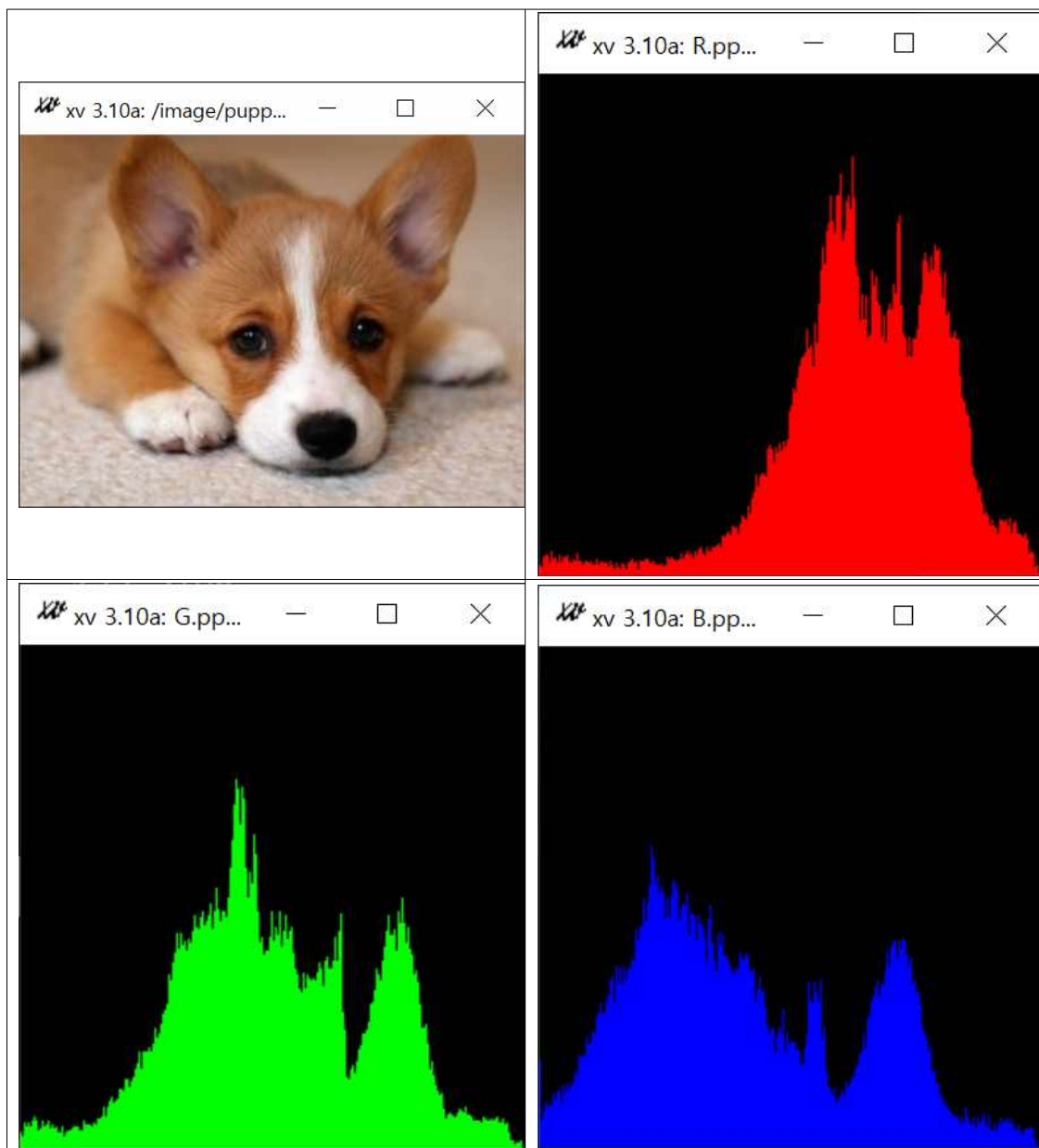
```



```

What are you doing?
[kimhyungho@raspberrry4 ~]gcc chist.c -o chist
[kimhyungho@raspberrry4 ~]xv /image/puppy.ppm &
[1] 2705
[kimhyungho@raspberrry4 ~]chist /image/puppy.ppm R.ppm 0
[kimhyungho@raspberrry4 ~]xv R.ppm &
[2] 2734
[kimhyungho@raspberrry4 ~]chist /image/puppy.ppm G.ppm 1
[kimhyungho@raspberrry4 ~]xv G.ppm &
[3] 2738
[kimhyungho@raspberrry4 ~]chist /image/puppy.ppm B.ppm 2
[kimhyungho@raspberrry4 ~]xv B.ppm &
[4] 2740
[kimhyungho@raspberrry4 ~]

```



fscanf로 읽기모드로 연 입력영상 /image/puppy.ppm의 헤더와 데이터 포맷을 읽어 들였습니다. 읽어 들일 때 컬러 영상은 한 픽셀에 값을 3개 가지므로 가로 픽셀 값을 읽어 들일 때 그레이 스케일 영상보다 3배 더 많이 읽어 들였습니다. 읽어 들인 픽셀 값의 빈도수를 체크하기 전에 argv[3](실행시 입력할 4번째 문자열)이 0이면 R히스토그램, 1이면 G히스토그램, 2면 B히스토그램을 출력할 수 있게 4번째 입력 문자열을 정수화 시켜 n에 넣어 설정했습니다. 정수화 과정에서 atoi함수를 쓰기 때문에 헤더파일 stdlib.h도 등록시켰습니다. 읽어 들인 R, G, B중 하나의 데이터 포맷의 픽셀 값을 반복문을 통해서 한번 씩 bright에 넣고 frequency[bright] = frequency[bright] + 1을 통해서 같은 픽셀 값이 나올 때 마다 빈도수를 증가 시켰습니다. image2[height][bright*3+n] = 255에서 bright에 3을 곱하고 n을 더해서 선택한 R, G, B에 맞게 히스토그램 픽셀 값 위치를 조정했습니다. 이후 (1)번과 같이 x축 대칭이동 시킨 후 컴파일하여 실행파일 chist를 만든 후 /image/puppy.ppm의 R, G, B별 컬러 히스토그램을 만들어 후면작업으로 입력이미지와 함께 출력했습니다.

(3)다음 영상을 밝게 만드는 프로그램을 작성하라. 입력영상과 출력 영상의 영상 히스토그램도 각각 그리고 분석하라.

입력영상: /image/puppy.pgm (20점)

```
#include <stdio.h>

int main(int argc, char * argv[]) //argc는 외부에서 입력한 문자열 개수,
{
    //argv[]는 외부에서 입력한 문자열들
    int image1[600][800]; //800x600 영상까지 처리가능
    int image2[600][800]; //800x600 영상까지 처리가능

    int x, y; //반복문에서 사용할 정수 변수 x, y 선언
    char M, N; //문자 변수 M과 N 선언
    int XX, YY, MAX; //정수 변수 XX, YY, MAX 선언
    FILE *file1, *file2; //파일 변수 file1과 file2 선언
    file1 = fopen(argv[1], "r"); //외부에서 입력한 두 번째 문자열을 읽기모드로 열
    fscanf(file1, "%c", &M); //헤더의 매직넘버를 읽어 들임 'P'
    fscanf(file1, "%c", &N); //헤더의 매직넘버를 읽어 들임 '2'
    fscanf(file1, "%d", &XX); //입력영상의 가로크기 읽어 들임
    fscanf(file1, "%d", &YY); //입력영상의 세로크기 읽어 들임
    fscanf(file1, "%d", &MAX); //입력영상의 최대 명암도 읽어 들임

    for(y = 0; y < YY; y++) //반복문을 사용하여 입력영상의 픽셀 값 읽어 들임
        for(x = 0; x < XX; x++)
            fscanf(file1, "%d", &image1[y][x]);

    for(y = 0; y < YY; y++)
        for(x = 0; x < XX; x++)
        {
```

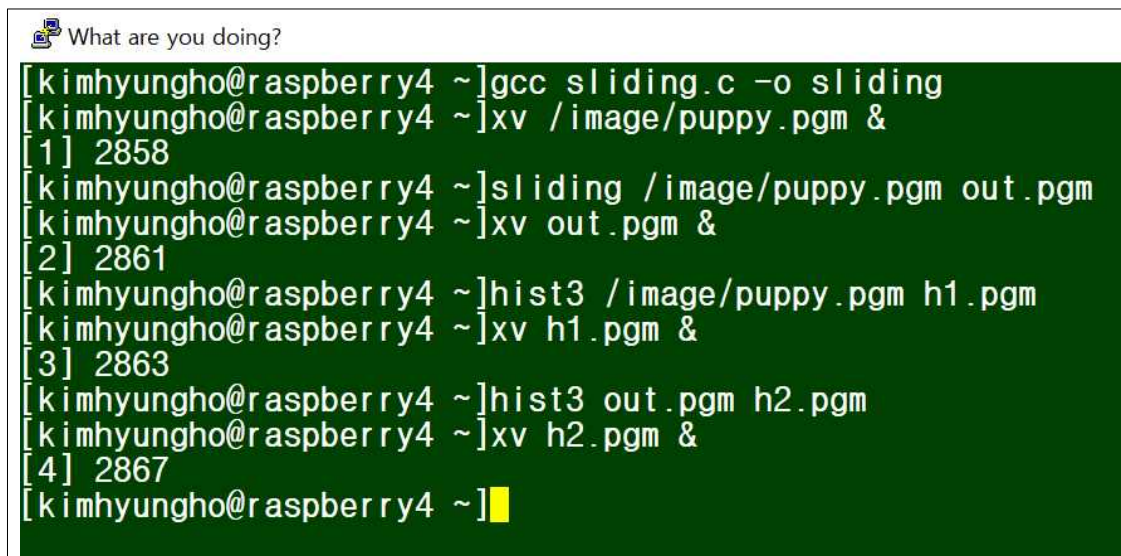
```

    image2[y][x] = image1[y][x] + 50; //픽셀 값에 50을 더해줘서 명암을 밝아지게 함
    if(image2[y][x] > 255)                //픽셀 값이 255가 넘으면 255로 고정
        image2[y][x] = 255;
    else if(image2[y][x] < 0)              //픽셀 값이 0보다 작으면 0으로 고정
        image2[y][x] = 0;
}

file2 = fopen(argv[2], "w");              //외부에서 입력한 세 번째 문자열을 쓰기모드로 열
fprintf(file2, "%c", M);                  //읽어 들인 두 번째 문자열 파일의 헤더 매직넘버를
fprintf(file2, "%cWn", N);                //세 번째 문자열 파일에 입력
fprintf(file2, "%d %dWn", XX, YY);        //읽어 들인 가로축과 세로축 크기를 입력
fprintf(file2, "%dWn", MAX);              //읽어 들인 최대 명암도를 입력

for(y = 0; y < YY; y++){
    for(x = 0; x < XX; x++){               //반복문을 이용하여 밝게 만든 픽셀 값을
        fprintf(file2, "%4d", image2[y][x]); //세 번째 문자열 파일에 입력
    }
    fprintf(file2, "Wn");                  //줄 내림 입력
}
return 0;
}

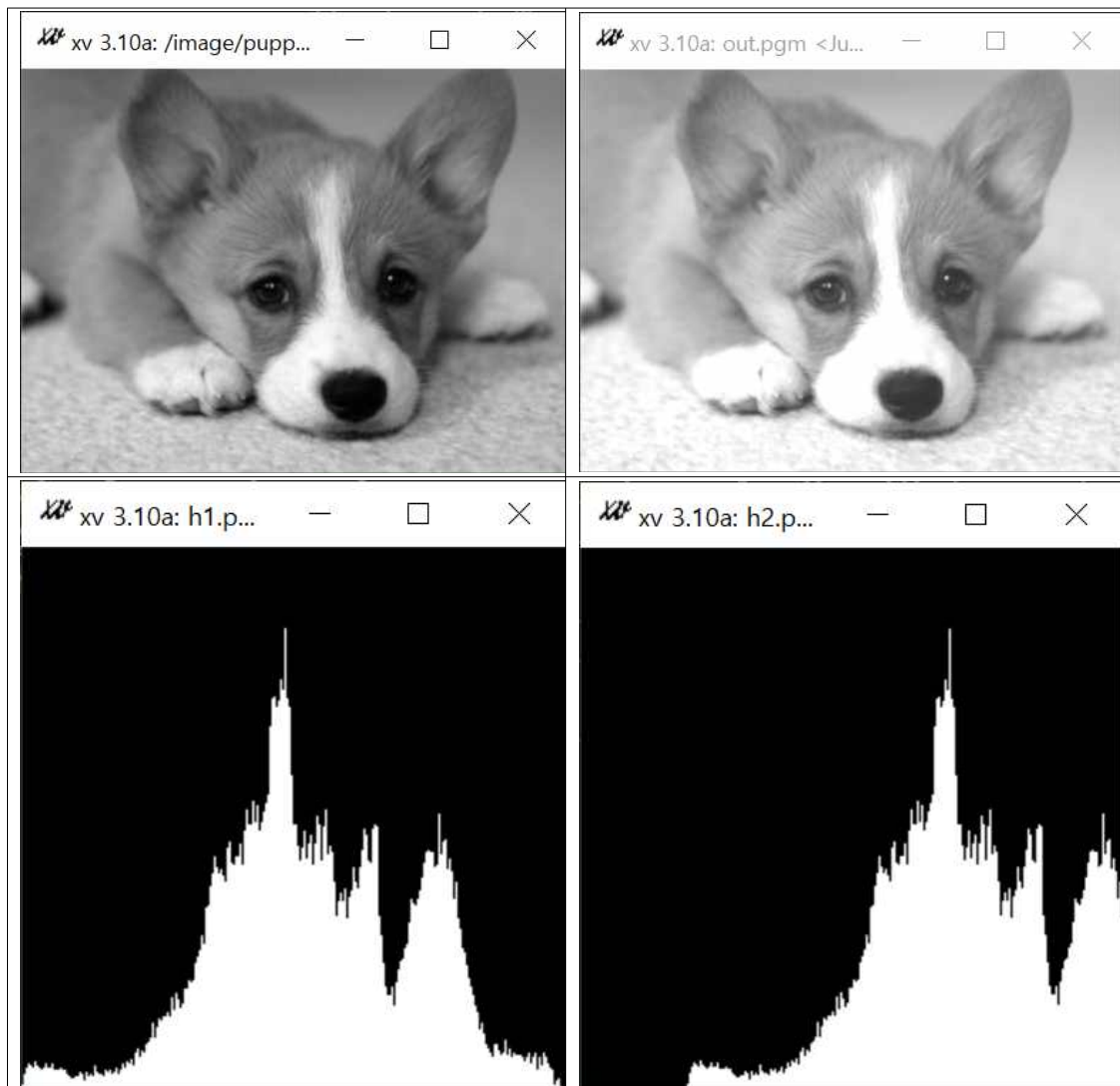
```



```

What are you doing?
[kimhyungho@raspberrry4 ~]gcc sliding.c -o sliding
[kimhyungho@raspberrry4 ~]xv /image/puppy.pgm &
[1] 2858
[kimhyungho@raspberrry4 ~]sliding /image/puppy.pgm out.pgm
[kimhyungho@raspberrry4 ~]xv out.pgm &
[2] 2861
[kimhyungho@raspberrry4 ~]hist3 /image/puppy.pgm h1.pgm
[kimhyungho@raspberrry4 ~]xv h1.pgm &
[3] 2863
[kimhyungho@raspberrry4 ~]hist3 out.pgm h2.pgm
[kimhyungho@raspberrry4 ~]xv h2.pgm &
[4] 2867
[kimhyungho@raspberrry4 ~]

```

fscanf로 읽기모드로 연 입력영상 /image/puppy.pgm의 헤더와 데이터 포맷을 읽어 들였습니다. $image2[y][x] = image1[y][x] + 50$ 에서 입력 영상의 픽셀 값들에 50을 더 해서 더 밝은 픽셀 값을 출력 영상으로 쓰일 image2에 넣었고 만약 픽셀 값이 255보다 크면 255로 고정되게 0보다 작으면 0에 고정되게 설정했습니다. fprintf로 쓰기모드로 연 out.pgm에 읽어 들인 입력영상의 헤더파일과 아까 밝게 만들었던 픽셀 값을 가진 image2 배열을 반복문을 통해서 입력했습니다. 이 후 실행과정에서 컴파일한 sliding.c의 실행파일을 sliding으로 만들었고 sliding을 통해 입력영상으로 /image/puppy.pgm을 넣고 출력영상으로 out.pgm을 설정했습니다. 그리고 xv와 후면 작업 &를 통해서 입력영상과 입력영상을 밝게 만든 out.pgm을 동시에 출력했고 (1)번에서 작업했던 hist3을 통해 출력영상과 입력영상의 히스토그램까지 동시에 출력했습니다.

(4) (3)번 문제를 어렵게 만드는 프로그램으로 해서 반복해라 (20점)

```
#include <stdio.h>

int main(int argc, char * argv[]) //argc는 외부에서 입력한 문자열 개수,
{ //argv[]는 외부에서 입력한 문자열들
    int image1[600][800]; //800x600 영상까지 처리가능
    int image2[600][800]; //800x600 영상까지 처리가능

    int x, y; //반복문에서 사용할 정수 변수 x, y 선언
    char M, N; //문자 변수 M과 N 선언
    int XX, YY, MAX; //정수 변수 XX, YY, MAX 선언
    FILE *file1, *file2; //파일 변수 file1과 file2 선언
    file1 = fopen(argv[1], "r"); //외부에서 입력한 두 번째 문자열을 읽기모드로 열
    fscanf(file1, "%c", &M); //헤더의 매직넘버를 읽어 들임 'P'
    fscanf(file1, "%c", &N); //헤더의 매직넘버를 읽어 들임 '2'
    fscanf(file1, "%d", &XX); //입력영상의 가로크기 읽어 들임
    fscanf(file1, "%d", &YY); //입력영상의 세로크기 읽어 들임
    fscanf(file1, "%d", &MAX); //입력영상의 최대 명암도 읽어 들임

    for(y = 0; y < YY; y++) //반복문을 사용하여 입력영상의 픽셀 값 읽어 들임
        for(x = 0; x < XX; x++)
            fscanf(file1, "%d", &image1[y][x]);

    for(y = 0; y < YY; y++)
        for(x = 0; x < XX; x++)
        {
            image2[y][x] = image1[y][x] - 60; //픽셀 값에 60을 빼줘서 명암을 어두워지게 함
            if(image2[y][x] > 255) //픽셀 값이 255가 넘으면 255로 고정
                image2[y][x] = 255;
            else if(image2[y][x] < 0) //픽셀 값이 0보다 작으면 0으로 고정
                image2[y][x] = 0;
        }

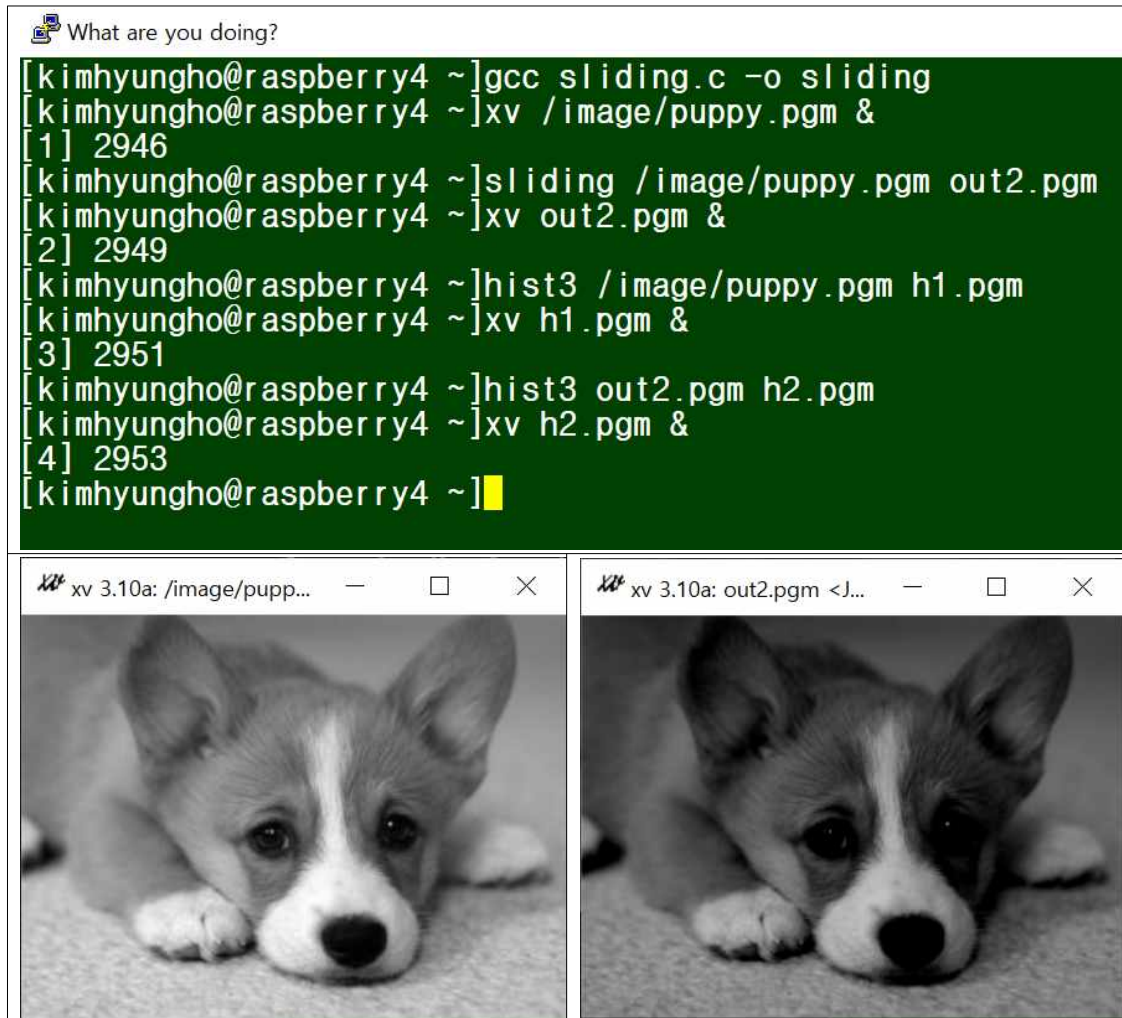
    file2 = fopen(argv[2], "w"); //외부에서 입력한 세 번째 문자열을 쓰기모드로 열
    fprintf(file2, "%c", M); //읽어 들인 두 번째 문자열 파일의 헤더 매직넘버를
    fprintf(file2, "%cWn", N); //세 번째 문자열 파일에 입력
    fprintf(file2, "%d %dWn", XX, YY); //읽어 들인 가로축과 세로축 크기를 입력
    fprintf(file2, "%dWn", MAX); //읽어 들인 최대 명암도를 입력

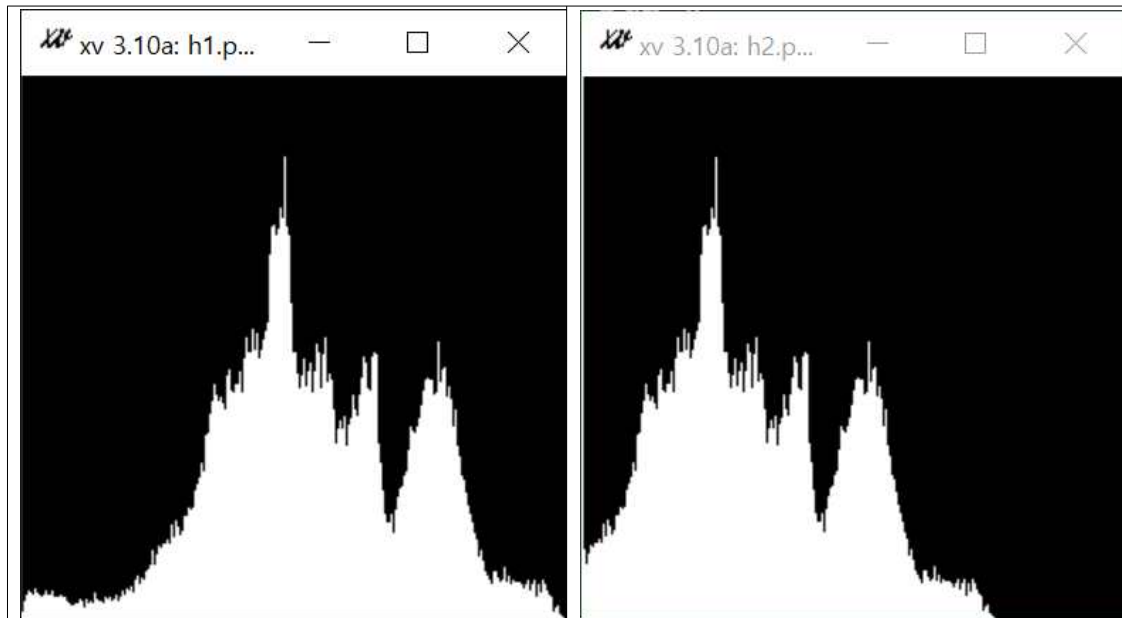
    for(y = 0; y < YY; y++){
        for(x = 0; x < XX; x++){ //반복문을 이용하여 어렵게 만든 픽셀 값을
            fprintf(file2, "%4d", image2[y][x]); //세 번째 문자열 파일에 입력
```

```

}
fprintf(file2, "Wn");           //줄 내림 입력
}
return 0;
}

```





fscanf로 읽기모드로 연 입력영상 /image/puppy.pgm의 헤더와 데이터 포맷을 읽어 들였습니다. $image2[y][x] = image1[y][x] - 60$ 에서 입력 영상의 픽셀 값들에 60을 빼서 더 어두운 픽셀 값을 출력 영상으로 쓰일 image2에 넣었고 만약 픽셀 값이 255보다 크면 255로 고정되게 0보다 작으면 0에 고정되게 설정했습니다. fprintf로 쓰기모드로 연 out2.pgm에 읽어 들인 입력영상의 헤더파일과 아까 밝게 만들었던 픽셀 값을 가진 image2 배열을 반복문을 통해서 입력했습니다. 이 후 실행과정에서 컴파일한 sliding.c의 실행파일을 sliding으로 만들었고 sliding을 통해 입력영상으로 /image/puppy.pgm을 넣고 출력영상으로 out2.pgm을 설정했습니다. 그리고 xv와 후면 작업 &를 통해서 입력영상과 입력영상을 어둡게 만든 out2.pgm을 동시에 출력했고 (1)번에서 작업했던 hist3을 통해 출력영상과 입력영상의 히스토그램까지 동시에 출력했습니다.

(5) 임의의 컬러 영상(각자 선택)을 밝게(10점) 그리고 어둡게(10점) 만드는 프로그램을 작성하라.

```
#include <stdio.h>

int main(int argc, char * argv[]) //argc는 외부에서 입력한 문자열 개수,
{
    //argv[]는 외부에서 입력한 문자열들
    int image1[600][800*3]; //800x600 영상까지 처리가능 컬러영상
    int image2[600][800*3]; //800x600 영상까지 처리가능 컬러영상

    int x, y; //반복문에서 사용할 정수 변수 x, y 선언
    char M, N; //문자 변수 M과 N 선언
    int XX, YY, MAX; //정수 변수 XX, YY, MAX 선언
    FILE *file1, *file2; //파일 변수 file1과 file2 선언
    file1 = fopen(argv[1], "r"); //외부에서 입력한 두 번째 문자열을 읽기모드로 옴
    fscanf(file1, "%c", &M); //헤더의 매직넘버를 읽어 들임 'P'
    fscanf(file1, "%c", &N); //헤더의 매직넘버를 읽어 들임 '3'
    fscanf(file1, "%d", &XX); //입력영상의 가로크기 읽어 들임
    fscanf(file1, "%d", &YY); //입력영상의 세로크기 읽어 들임
    fscanf(file1, "%d", &MAX); //입력영상의 최대 명암도 읽어 들임

    for(y = 0; y < YY; y++) //반복문을 사용하여 입력영상의 픽셀 값 읽어 들임
        for(x = 0; x < XX*3; x++) //컬러영상은 가로픽셀 당 값을 3개 가지므로 x3
            fscanf(file1, "%d", &image1[y][x]);

    for(y = 0; y < YY; y++)
        for(x = 0; x < XX*3; x++) //컬러영상은 가로픽셀 당 값을 3개 가지므로 x3
        {
            //픽셀 값에 100을 더해줘서 명암을 밝아지게 함
            image2[y][x] = image1[y][x] + 100;
            if(image2[y][x] > 255)
                image2[y][x] = 255; //픽셀 값이 255가 넘으면 255로 고정
            else if(image2[y][x] < 0)
                image2[y][x] = 0; //픽셀 값이 0보다 작으면 0으로 고정
        }

    file2 = fopen(argv[2], "w"); //외부에서 입력한 세 번째 문자열을 쓰기모드로 옴
    fprintf(file2, "%c", M); //읽어 들인 두 번째 문자열 파일의 헤더 매직넘버를
    fprintf(file2, "%cWn", N); //세 번째 문자열 파일에 입력
    fprintf(file2, "%d %dWn", XX, YY); //읽어 들인 가로축과 세로축 크기를 입력
    fprintf(file2, "%dWn", MAX); //읽어 들인 최대 명암도를 입력

    for(y = 0; y < YY; y++){
        for(x = 0; x < XX*3; x++){ //반복문을 이용하여 밝게 만든 픽셀 값을
```

```

    fprintf(file2,"%4d", image2[y][x]); //세 번째 문자열 파일에 입력
}
fprintf(file2, "\n");    //줄 내림 입력
}
return 0;
}

```

What are you doing?

```

[kimhyungho@raspberrypi4 ~]gcc cslicing.c -o cslicing
[kimhyungho@raspberrypi4 ~]xv totoro.ppm &
[1] 4501
[kimhyungho@raspberrypi4 ~]unlimit
[kimhyungho@raspberrypi4 ~]cslicing totoro.ppm out.ppm
[kimhyungho@raspberrypi4 ~]xv out.ppm &
[2] 4505
[kimhyungho@raspberrypi4 ~]

```

xv 3.10a: totoro.ppm <Jung, Minchul>



xv 3.10a: out.ppm <Jung, Minchul>



먼저 cmd에서 sftp와 put 명령어를 사용하여 totoro.jpg를 라즈베리파이 서버에 넣은 뒤 xming에서 ppm으로 변환했습니다. 그리고 fscanf로 읽기모드로 연 입력영상 totoro.ppm의 헤더와 데이터 포맷을 읽어 들였습니다. 읽어 들일 때 컬러 영상은 한 픽셀에 값을 3개 가지므로 가로 픽셀 값을 읽어 들일 때 그레이 스케일 영상보다 3배 더 많이 읽어 들였습니다. $image2[y][x] = image1[y][x] + 100$ 을 통해 입력 영상의 픽셀 값들에 100을 더해서 더 밝은 픽셀 값을 출력 영상으로 쓰일 image2에 넣었고 여기서도 컬러이미지 이기 때문에 곱하기 3을 해줬습니다. 만약 픽셀 값이 255보다 크면 255로 고정되게 0보다 작으면 0에 고정되게 설정했습니다. fprintf로 쓰기모드로 연 out.ppm에 읽어 들인 입력영상의 헤더파일과 아까 밝게 만들었던 픽셀 값을 가진 image2 배열을 반복문을 통해서 입력했습니다. 이 후 실행과정에서 컴파일한 cslicing.c의 실행파일을 cslicing으로 만들었고 cslicing을 통해 입력영상 totoro.ppm을 넣고 출력영상으로 out.ppm을 설정했습니다. 그리고 xv와 후면 작업 &를 통해서 입력영상과 출력영상을 밝게 만든 out.ppm을 동시에 출력했습니다.

```

#include <stdio.h>
int main(int argc, char * argv[]) //argc는 외부에서 입력한 문자열 개수,
{
    //argv[]는 외부에서 입력한 문자열들
    int image1[600][800*3]; //800x600 영상까지 처리가능 컬러영상
    int image2[600][800*3]; //800x600 영상까지 처리가능 컬러영상

    int x, y; //반복문에서 사용할 정수 변수 x, y 선언
    char M, N; //문자 변수 M과 N 선언
    int XX, YY, MAX; //정수 변수 XX, YY, MAX 선언
    FILE *file1, *file2; //파일 변수 file1과 file2 선언
    file1 = fopen(argv[1], "r"); //외부에서 입력한 두 번째 문자열을 읽기모드로 옴
    fscanf(file1, "%c", &M); //헤더의 매직넘버를 읽어 들임 'P'
    fscanf(file1, "%c", &N); //헤더의 매직넘버를 읽어 들임 '3'
    fscanf(file1, "%d", &XX); //입력영상의 가로크기 읽어 들임
    fscanf(file1, "%d", &YY); //입력영상의 세로크기 읽어 들임
    fscanf(file1, "%d", &MAX); //입력영상의 최대 명암도 읽어 들임

    for(y = 0; y < YY; y++) //반복문을 사용하여 입력영상의 픽셀 값 읽어 들임
        for(x = 0; x < XX*3; x++) //컬러영상은 가로픽셀 당 값을 3개 가지므로 x3
            fscanf(file1, "%d", &image1[y][x]);

    for(y = 0; y < YY; y++)
        for(x = 0; x < XX*3; x++) //컬러영상은 가로픽셀 당 값을 3개 가지므로 x3
        {
            //픽셀 값에 100을 빼줘서 명암을 어두워지게 함
            image2[y][x] = image1[y][x] - 100;
            if(image2[y][x] > 255)
                image2[y][x] = 255; //픽셀 값이 255가 넘으면 255로 고정
            else if(image2[y][x] < 0)
                image2[y][x] = 0; //픽셀 값이 0보다 작으면 0으로 고정
        }

    file2 = fopen(argv[2], "w"); //외부에서 입력한 세 번째 문자열을 쓰기모드로 옴
    fprintf(file2, "%c", M); //읽어 들인 두 번째 문자열 파일의 헤더 매직넘버를
    fprintf(file2, "%cWn", N); //세 번째 문자열 파일에 입력
    fprintf(file2, "%d %dWn", XX, YY); //읽어 들인 가로축과 세로축 크기를 입력
    fprintf(file2, "%dWn", MAX); //읽어 들인 최대 명암도를 입력

    for(y = 0; y < YY; y++){
        for(x = 0; x < XX*3; x++){ //반복문을 이용하여 어둡게 만든 픽셀 값을
            fprintf(file2, "%4d", image2[y][x]); //세 번째 문자열 파일에 입력
        }
    }
}

```



```

printf(file2, "Wn");    //줄 내림 입력
}
return 0;
}

```



```

What are you doing?
[kimhyungho@raspberrypi ~]$ gcc cslliding.c -o cslliding
[kimhyungho@raspberrypi ~]$ xv totoro.ppm &
[1] 4531
[kimhyungho@raspberrypi ~]$ cslliding totoro.ppm out.ppm
[kimhyungho@raspberrypi ~]$ xv out.ppm &
[2] 4533
[kimhyungho@raspberrypi ~]$

```



먼저 cmd에서 sftp와 put 명령어를 사용하여 totoro.jpg를 라즈베리파이 서버에 넣은 뒤 xming에서 ppm으로 변환했습니다. 그리고 fscanf로 읽기모드로 연 입력영상 totoro.ppm의 헤더와 데이터 포맷을 읽어 들였습니다. 읽어 들일 때 컬러 영상은 한 픽셀에 값을 3개 가지므로 가로 픽셀 값을 읽어 들일 때 그레이 스케일 영상보다 3배 더 많이 읽어 들였습니다. $image2[y][x] = image1[y][x] - 100$ 을 통해 입력 영상의 픽셀 값들에 100을 빼서 더 어두운 픽셀 값을 출력 영상으로 쓰일 image2에 넣었고 여기서도 컬러이미지 이기 때문에 곱하기 3을 해줬습니다. 만약 픽셀 값이 255보다 크면 255로 고정되게 0보다 작으면 0에 고정되게 설정했습니다. printf로 쓰기모드로 연 out.ppm에 읽어 들인 입력영상의 헤더파일과 아까 어둡게 만들었던 픽셀 값을 가진 image2 배열을 반복문을 통해서 입력했습니다. 이 후 실행과정에서 컴파일한 cslliding.c의 실행파일을 cslliding으로 만들었고 cslliding을 통해 입력영상 totoro.ppm을 넣고 출력영상으로 out.ppm을 설정했습니다. 그리고 xv와 후면 작업 &를 통해서 입력영상과 입력영상을 밝게 만든 out.ppm을 동시에 출력했습니다.

