```c
//read_pgm.c

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

#define MAX_HEIGHT 1025
#define MAX_WIDTH 1025

#define MAX_PIXELS_PER_LINE 70

#define MIN( x, y ) ( ( x ) < ( y ) ? ( x ) : ( y ) )

int theImageArray[1025][MAX_WIDTH];
int theImageArrayDup[1025][MAX_WIDTH];

FILE* read_pgm_file_info
(
   int* height,
   int* width,
   int* maxPixel,
   char* pgmFname
)
{
   static FILE* fp = 0L ;
   char pgmFormatFlag[3] =  { '\0' } ;

   char trash[80] = { '\0' } ;
   memset ( ( void * ) trash, '\0', 80 ) ;

   fp = fopen ( pgmFname, "r" ) ;

   if ( fp )
   {

       fscanf ( fp, "%2c\n", pgmFormatFlag ) ;

       if ( ! strcmp ( pgmFormatFlag, "P2" ) )
       {
```

```
            fgets (  trash, 70, fp ) ;

            fscanf ( fp, "%i", width ) ;
             fscanf ( fp, "%i", height ) ;

            fscanf ( fp, "%i", maxPixel ) ;
        }

    }

    return fp ;
}

void write_pgm_file_from_array
(
    char* pgmOutFileName,
    int imageArray[][MAX_WIDTH],
    char* commentLine,
    int height,
    int width,
    int maxPixel
)
{
    int row = 0 ;
    int col = 0 ;
    FILE* fp = fopen ( pgmOutFileName, "w" ) ;
    if ( fp )
    {

        fprintf ( fp, "P2\n" ) ;
        fprintf ( fp, "%s\n", commentLine ) ;
        fprintf ( fp, "%u %u\n", width, height ) ;
        fprintf ( fp, "%u\n", maxPixel ) ;

        for ( row = 0 ; row < height ; row ++ )
        {
            for ( col = 0 ; col < width ; col ++ )
            {
                fprintf ( fp, "%u", imageArray[row][col] ) ;
```

```
                if ( MAX_PIXELS_PER_LINE > 1 )
                {
                    fprintf ( fp, " " ) ;
                }
                if ( ( col % MAX_PIXELS_PER_LINE ) == 0 )
                {
                    fprintf ( fp, "\n" ) ;
                }
            }

            if ( col % MAX_PIXELS_PER_LINE )
            {
                fprintf ( fp, "\n" ) ;
            }
        }
        fclose ( fp ) ;
    }
    return ;
}

void read_pgm_file_into_array
(
    int imageArray[][MAX_WIDTH],
    int* height,
    int* width,
    int* maxPixel,
    char* pgmInFileName
)
{
    int row = 0 ;
    int col = 0 ;
    FILE* fp = read_pgm_file_info ( height, width, maxPixel, pgmInFileName
) ;
    char trash = ' ';
    char yesThreshold = ' ';

    if ( fp )
    {
        printf ( "reading height=%d, width=%d\n", *height, *width ) ;
```

```
        for ( row = 0 ; row < MIN( MAX_HEIGHT - 1, *height ) ; row ++ )
        {
            for ( col = 0 ; col < MIN( MAX_WIDTH -1, *width ) ; col ++ )
            {
                fscanf ( fp, "%i", &imageArray[row][col]) ;
            }
        }
        fclose ( fp ) ;
    }

    return ;
}

int main( void )
{
    int height  = 0 ;
    int width = 0 ;
    int maxPixel = 0 ;

    char* pgmInFileName = "irv.pgm" ;
    char* pgmOutFileName = "irv_out.pgm" ;

    read_pgm_file_into_array (theImageArray, &height, &width, &maxPixel,
pgmInFileName ) ;

    for(int i = 0; i < 1025; i++){
        for(int j = 0; j < MAX_WIDTH; j++){
            theImageArrayDup[i][j] = theImageArray[i][j];
        }
    }
    imageProcessing(theImageArray, theImageArrayDup,height,width);

    write_pgm_file_from_array ( pgmOutFileName, theImageArray, "# JR test
file", height, width,
                                maxPixel ) ;

    printf ( "Copying %s to %s, height=%u, width=%u, maxPixel=%d\n",
pgmInFileName,
             pgmOutFileName,
             height, width, maxPixel ) ;
```

```
   return 0 ;
}
```

```
//function_call.c

#define MAX_HEIGHT 1025
#define MAX_WIDTH 1025
#include <stdio.h>
#include <math.h>
#include <string.h>


void convolve(int theImageArray[1025][1025], int
theImageArrayDup[1025][1025], int height, int width){
   int boxBlur[3][3] = {
       { 1, 1, 1 },
       { 1,  1, 1 },
       { 1, 1, 1 }
   };

   int gaussian[3][3] = {
       {1, 2, 1},
       {2, 4, 2},
       {1, 2, 1}
   };
    int edgeDetect[3][3] = {
       {-1, -1, -1},
       {-1,  5, -1},
       {-1, -1, -1}
   };

   int sharpen[3][3] = {
       {0, -1,  0},
       {-1, 5, -1},
       {0, -1,  0}
   };
```

```
   char convolution[30];
   printf("Would you like to run box blur, Gaussian blur, edge detection,
or sharpen?\n(Box, Gaussian, Edge, Sharpen) \n");
   scanf("%s", convolution);
   lowerletter(convolution);

   if(strcmp(convolution, "box") == 0){
       for (int i = 1; i < MAX_HEIGHT; i++) {
           for (int j = 1; j < MAX_WIDTH; j++) {
               conMatrixMult(theImageArray, theImageArrayDup, boxBlur, i,
j, 9, 0);

           }
       }
   }
   if(strcmp(convolution, "gaussian") == 0){
       for (int i = 1; i < MAX_HEIGHT; i++) {
           for (int j = 1; j < MAX_WIDTH; j++) {
               conMatrixMult(theImageArray, theImageArrayDup, gaussian, i,
j, 16, 0);

           }
       }
   }
   if(strcmp(convolution, "edge") == 0){
       for (int i = 1; i < MAX_HEIGHT; i++) {
           for (int j = 1; j < MAX_WIDTH; j++) {
               conMatrixMult(theImageArray, theImageArrayDup, edgeDetect,
i, j, 128, 0);

           }
       }
   }
   if(strcmp(convolution, "sharpen") == 0){
       for (int i = 1; i < MAX_HEIGHT; i++) {
           for (int j = 1; j < MAX_WIDTH; j++) {
               conMatrixMult(theImageArray, theImageArrayDup, sharpen, i,
j, 8, -50);

               theImageArray[i][j] = 2*(theImageArray[i][j] -50) + 100;
```

```
                if (theImageArray[i][j] > 255) {
                    theImageArray[i][j] = 255;
                }
            }

        }
   printf("Image is washed out.");
   }
}

void conMatrixMult(int theImageArray[1025][1025], int
theImageArrayDup[1025][1025], int myMatrix[3][3], int i, int j, int
divisor, int subtractor){
    theImageArray[i][j] = (theImageArray[i+1][j+1]*myMatrix[0][0] +
theImageArray[i][j+1]*myMatrix[0][1]+
theImageArray[i-1][j+1]*myMatrix[0][2]+theImageArray[i+1][j]*myMatrix[1][0
]+theImageArray[i][j]*myMatrix[1][1]+theImageArray[i-1][j]*myMatrix[1][2]+
theImageArray[i+1][j-1]*myMatrix[2][0]+theImageArray[i][j+1]*myMatrix[2][1
]+theImageArray[i-1][j-1]*myMatrix[2][2])/divisor - subtractor;
}

void invert(int theImageArray[1025][1025]){
   for (int i = 1; i < MAX_HEIGHT; i++) {
       for (int j = 1; j < MAX_WIDTH; j++) {
             theImageArray[i][j] = 255 - theImageArray[i][j];
       }
   }

}

void threshold(int theImageArray[1025][1025]) {


   int n;
   printf("Enter the number of levels you wish to threshold by. \n(Integer
between 2-6 inclusive). \n");
   scanf("%d", &n);

   for (int i = 0; i < MAX_HEIGHT; i++) {
       for (int j = 0; j < MAX_WIDTH; j++) {
```

```
        switch (n){

        case 1:
            printf("Invalid response. Please start over. /n");
            break;

        case 2:
            if (theImageArray[i][j] > (n-1)*255/n) {
                theImageArray[i][j] = 255;
            }
            else {
                theImageArray[i][j] = 0;
            }
            break;
        case 3:
            if (theImageArray[i][j] > (n-1)*255/n) {
                theImageArray[i][j] = 255;
            }
            else if (theImageArray[i][j] > (n-2)*(255/n)){
                theImageArray[i][j] = (n-2)*(255/n);
            }
            else {
                theImageArray[i][j] = 0;
            }
            break;
        case 4:
            if (theImageArray[i][j] > (n-1)*255/n) {
                theImageArray[i][j] = 255;
            }
            else if (theImageArray[i][j] > (n-2)*(255/n)){
                theImageArray[i][j] = (n-2)*(255/n);
            }
            else if (theImageArray[i][j] > (n-3)*(255/n)){
                theImageArray[i][j] = (n-3)*(255/n);
            }
            else {
                theImageArray[i][j] = 0;
            }
            break;
        case 5:
```

```
            if (theImageArray[i][j] > (n-1)*255/n) {
                theImageArray[i][j] = 255;
            }
            else if (theImageArray[i][j] > (n-2)*(255/n)){
                theImageArray[i][j] = (n-2)*(255/n);
            }
            else if (theImageArray[i][j] > (n-3)*(255/n)){
                theImageArray[i][j] = (n-3)*(255/n);
            }
            else if (theImageArray[i][j] > (n-4)*(255/n)){
                theImageArray[i][j] = (n-4)*(255/n);
            }
            else {
                theImageArray[i][j] = 0;
            }
            break;
        case 6:
            if (theImageArray[i][j] > (n-1)*255/n) {
                theImageArray[i][j] = 255;
            }
            else if (theImageArray[i][j] > (n-2)*(255/n)){
                theImageArray[i][j] = (n-2)*(255/n);
            }
            else if (theImageArray[i][j] > (n-3)*(255/n)){
                theImageArray[i][j] = (n-3)*(255/n);
            }
            else if (theImageArray[i][j] > (n-4)*(255/n)){
                theImageArray[i][j] = (n-4)*(255/n);
            }
            else if (theImageArray[i][j] > (n-5)*(255/n)){
                theImageArray[i][j] = (n-5)*(255/n);
            }
            else {
                theImageArray[i][j] = 0;
            }
            break;
        default:
            printf("Invalid response. Please start over. /n");
            break;
        }
```

```
        }
    }
}

void luminosity(int theImageArray[1025][1025]) {
    char lum[10];
    double perc;
    printf("Do you want to lighten or darken the image? (Lighten, Darken)
\n");
    scanf("%s", lum);
    lowerletter(lum);

    printf("Enter a percentage between 0 and 100 (without the %%) of how
much you want\nthe luminosity to change by. \n");
    scanf("%lf", &perc);
    perc = 0.01* perc;


    if(strcmp(lum, "lighten") == 0){
        perc = 1.0 + perc;

        for (int i = 0; i < MAX_HEIGHT; i++) {
            for (int j = 0; j < MAX_WIDTH; j++) {
                if(perc*theImageArray[i][j]<= 255){
                    theImageArray[i][j] = perc*theImageArray[i][j];
                }
                else{
                    theImageArray[i][j] = 255;
                }
            }
        }
    }

    if(strcmp(lum, "darken") == 0){
        perc = 1.0 - perc;

        for (int i = 0; i < MAX_HEIGHT; i++) {
            for (int j = 0; j < MAX_WIDTH; j++) {
                theImageArray[i][j] = perc*theImageArray[i][j];
```

```
            }
        }
    }
}

void frame(int theImageArray[1025][1025], int width, int height){
    int startPosX;
    int endPosX;
    int startPosY;
    int endPosY;
    int lum;

    printf("Enter leftmost X position (0-%d): ", width);
    scanf("%d", &startPosX);
    printf("\nEnter rightmost X position (0-%d): ", width);
    scanf("%d", &endPosX);
    printf("\nEnter topmost Y position (0-%d): ", height);
    scanf("%d", &startPosY);
    printf("\nEnter bottommost Y position (0-%d): ", height);
    scanf("%d", &endPosY);
    printf("\n");
    printf("Enter frame luminosity (0-255): ");
    scanf("%d", &lum);
    printf("\n");

    for(int i = 0; i < MAX_WIDTH; i++){
        for(int j = 0; j < startPosY; j++){
            theImageArray[i][j] = lum;
        }
    }
    for(int i = 0; i < startPosX; i++){
        for(int j = startPosY; j < MAX_HEIGHT; j++){
            theImageArray[i][j] = lum;
        }
    }
    for(int i = startPosX; i < endPosX; i++){
        for(int j = endPosY; j < MAX_HEIGHT; j++){
            theImageArray[i][j] = lum;
        }
    }
```

```
   for(int i = endPosX; i < MAX_WIDTH; i++){
       for(int j = startPosY; j < MAX_HEIGHT; j++){
           theImageArray[i][j] = lum;
       }
   }

}

void transformation(int theImageArray[1025][1025], int
theImageArrayDup[1025][1025], int width, int height){
   char tfm[10];
   int rotation;
   printf("Do you want to rotate the image or reflect the image? (Rotate,
Reflect) \n");
   scanf("%s", tfm);
   lowerletter(tfm);

   if (strcmp(tfm, "reflect") == 0) {
       printf("Do you want the image to be reflected vertically or
horizontally?\n(Vertical, Horizontal)\n");
       char direction[15];
       scanf("%s", direction);
       lowerletter(direction);


       if(strcmp(direction, "vertical") == 0){
           for(int i = 0; i < width; i++){
               for(int j = 0; j < height; j++){
                   theImageArrayDup[i][j] = theImageArray[width-i][j];
               }
           }

           for(int i = 0; i < width; i++){
               for(int j = 0; j < height; j++){
                   theImageArray[i][j] = theImageArrayDup[i][j];
               }
           }
       }

       if(strcmp(direction,"horizontal") == 0){
```

```
            for(int i = 0; i < width; i++){
                for(int j = 0; j < height; j++){
                    theImageArrayDup[i][j] = theImageArray[i][height-j];
                }
            }

            for(int i = 0; i < width; i++){
                for(int j = 0; j < height; j++){
                    theImageArray[i][j] = theImageArrayDup[i][j];
                }
            }
        }
    }

    if(strcmp(tfm, "rotate") == 0) {
        printf("Do you want the image rotated 90, 180, or 270 degrees
clockwise? ");
        scanf("%d", &rotation);

        if (rotation == 90){
            for(int i = 0; i < MAX_WIDTH; i++){
                for(int j = 0; j < MAX_HEIGHT; j++){
                    theImageArrayDup[i][j] = theImageArray[j][i];
                }
            }

            for(int i = 0; i < MAX_WIDTH; i++){
                for(int j = 0; j < MAX_HEIGHT; j++){
                    theImageArray[i][j] = theImageArrayDup[i][j];
                }
            }
        }

         if(rotation ==  180) {
             for(int i = 0; i < width; i++){
                for(int j = 0; j < height; j++){
                    theImageArrayDup[i][j] =
theImageArray[width-i][height-j];
                }
            }
```

```
            for(int i = 0; i < width; i++){
                for(int j = 0; j < height; j++){
                    theImageArray[i][j] = theImageArrayDup[i][j];
                }
            }
        }

        if (rotation == 270){
            for(int i = 0; i < width; i++){
                for(int j = 0; j < height; j++){
                    theImageArrayDup[i][j] = theImageArray[height-j][i];
                }
            }

            for(int i = 0; i < width; i++){
                for(int j = 0; j < height; j++){
                    theImageArray[i][j] = theImageArrayDup[i][j];
                }
            }
        }
    }
}
```

```
//images.c

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

void imageProcessing(int theImageArray[1025][1025], int
theImageArrayDup[1025][1025], int height, int width) {
    char inputFunc[30];
```

```
   printf("Enter the function you wish to apply to your image. (Threshold,
Luminosity,\nFraming, Transform, Convolve, Invert) \n");
   scanf("%s", inputFunc);
   lowerletter(inputFunc);

   if(strcmp(inputFunc,"invert") == 0) {
       invert(theImageArray);
   }

   if(strcmp(inputFunc,"convolve") == 0) {
       convolve(theImageArray, height, width);
   }

   if(strcmp(inputFunc, "threshold") == 0){
       threshold(theImageArray);
   }

   if(strcmp(inputFunc, "luminosity") == 0){
       luminosity(theImageArray);
   }

   if(strcmp(inputFunc, "framing") == 0){
       frame(theImageArray, height, width);
   }

   if(strcmp(inputFunc,"transform") == 0) {
       transformation(theImageArray, theImageArrayDup, height, width);
   }


   printf("\nDone\n");
}


void lowerletter(char myString[30]){
   for(int i = 0; i < strlen(myString); i++){
       myString[i] = tolower(myString[i]);
   }

}
```