Max Truong
EECS 22
2/1/2025

AutoTest Function Example:

```
EngPlaza.ppm was loaded successfully!
negative.ppm was saved successfully.
0 mismatching pixels (0.000%) identified in negative_diff.ppm.
negative.jpg was stored for viewing.
Negative tested!

EngPlaza.ppm was loaded successfully!
colorfilter.ppm was saved successfully.
0 mismatching pixels (0.000%) identified in colorfilter_diff.ppm.
colorfilter.jpg was stored for viewing.
Color Filter tested!

EngPlaza.ppm was loaded successfully!
edge.ppm was saved successfully.
0 mismatching pixels (0.000%) identified in edge_diff.ppm.
edge.jpg was stored for viewing.
Edge Detection tested!

EngPlaza.ppm was loaded successfully!
hmirror.ppm was saved successfully.
0 mismatching pixels (0.000%) identified in hmirror_diff.ppm.
hmirror.jpg was stored for viewing.
HMirror tested!

EngPlaza.ppm was loaded successfully!
bw.ppm was saved successfully.
0 mismatching pixels (0.000%) identified in bw_diff.ppm.
bw.jpg was stored for viewing.
Black & White tested!

EngPlaza.ppm was loaded successfully!
vflip.ppm was saved successfully.
0 mismatching pixels (0.000%) identified in vflip_diff.ppm.
vflip.jpg was stored for viewing.
VFlip tested!

EngPlaza.ppm was loaded successfully!
shuffle.ppm was saved successfully.
0 mismatching pixels (0.000%) identified in shuffle_diff.ppm.
shuffle.jpg was stored for viewing.
Shuffle tested!

EngPlaza.ppm was loaded successfully!
pixelate.ppm was saved successfully.
0 mismatching pixels (0.000%) identified in pixelate_diff.ppm.
pixelate.jpg was stored for viewing.
Pixelate tested!
```

CODE SUMMARY

1. Global Definitions and Constants

c
Copy
```
#define WIDTH  512
#define HEIGHT 288
#define SLEN    80
#define MENU_SIZE 13
```
Purpose: These constants define the image dimensions (WIDTH and HEIGHT), the maximum length of file names (SLEN), and the number of menu items (MENU_SIZE).

Why: Using constants ensures consistency and makes the code easier to maintain. For example, if the image size changes, you only need to update these constants.

2. Function Declarations

c
Copy
```
void PrintMenu(void);
int LoadImage(...);
int SaveImage(...);
void BlackNWhite(...);
void Negative(...);
void ColorFilter(...);
void Edge(...);
void HMirror(...);
void Shuffle(...);
void VFlip(...);
void Pixelate(...);
void AutoTest(...);
int menuPrompt(void);
void callMenuFunctions(...);
```
Purpose: These are function prototypes for all the operations your program can perform, such as loading/saving images, applying filters, and manipulating images.

Why: Declaring functions at the top allows the compiler to recognize them before they are defined, making the code modular and easier to read.

3. Main Function

c
Copy
```
int main(void) {
```

```
    unsigned char R[WIDTH][HEIGHT] = {0};
    unsigned char G[WIDTH][HEIGHT] = {0};
    unsigned char B[WIDTH][HEIGHT] = {0};
    int choice = 0;

    while (choice != 1) {
        choice = menuPrompt();
        if (choice != 1) {
            printf("Please load an image first.\n");
        }
    }

    callMenuFunctions(choice, R, G, B);

    while (choice != 12) {
        choice = menuPrompt();
        callMenuFunctions(choice, R, G, B);
    }

    return 0;
}
```

Purpose: The main function initializes the image arrays (R, G, B) and handles the program flow. It ensures the user loads an image first and then processes user choices until they exit.

Why: This structure ensures the program is user-friendly and prevents operations on an unloaded image.

4. Menu System

menuPrompt Function

c

Copy

```
int menuPrompt(void) {
    int choice = 0;
    for (int i = 0; i < MENU_SIZE; i++) {
        printf("%s\n", menu_options[i]);
    }
    printf("Please make your choice: ");
    scanf("%d", &choice);
    while(getchar() != '\n');  // Discard extra characters in input buffer
    return choice;
}
```

Purpose: Displays the menu options and prompts the user for a choice.

Why: This modular approach makes it easy to update the menu or add new options.

callMenuFunctions Function
c
Copy
```
void callMenuFunctions(int choice, unsigned char R[WIDTH][HEIGHT], unsigned char
G[WIDTH][HEIGHT], unsigned char B[WIDTH][HEIGHT]) {
    switch (choice) {
        case 1: LoadImage(...); break;
        case 2: SaveImage(...); break;
        case 3: BlackNWhite(...); break;
        ...
    }
}
```
Purpose: Executes the appropriate function based on the user's choice.

Why: Using a switch statement makes the code clean and scalable for adding new features.

5. Image Loading and Saving

LoadImage Function
c
Copy
```
int LoadImage(const char fname[SLEN], unsigned char R[WIDTH][HEIGHT], unsigned char
G[WIDTH][HEIGHT], unsigned char B[WIDTH][HEIGHT]) {
    FILE *File;
    char Type[SLEN];
    int Width, Height, MaxValue;
    ...
}
```
Purpose: Reads a PPM image file and loads its RGB data into the R, G, and B arrays.

Why: This function ensures the image is in the correct format and dimensions before processing.

SaveImage Function
c
Copy
```
int SaveImage(const char fname[SLEN], unsigned char R[WIDTH][HEIGHT], unsigned char
G[WIDTH][HEIGHT], unsigned char B[WIDTH][HEIGHT]) {
    FILE *File;
    ...
}
```
Purpose: Saves the processed image as a PPM file and optionally converts it to JPEG.

Why: This allows users to save their work and view it in a common format.

6. Image Processing Functions

BlackNWhite Function

c

Copy

```
void BlackNWhite(unsigned char R[WIDTH][HEIGHT], unsigned char G[WIDTH][HEIGHT],
unsigned char B[WIDTH][HEIGHT]) {
    int x, y;
    unsigned char average;
    for (y = 0; y < HEIGHT; y++) {
        for (x = 0; x < WIDTH; x++) {
            average = (R[x][y] + G[x][y] + B[x][y]) / 3;
            R[x][y] = G[x][y] = B[x][y] = average;
        }
    }
}
```

Purpose: Converts the image to grayscale by averaging the RGB values.

Why: This is a common image processing operation that simplifies the image.

Negative Function

c

Copy

```
void Negative(unsigned char R[WIDTH][HEIGHT], unsigned char G[WIDTH][HEIGHT], unsigned
char B[WIDTH][HEIGHT]) {
    int x, y;
    for (y = 0; y < HEIGHT; y++) {
        for (x = 0; x < WIDTH; x++) {
            R[x][y] = 255 - R[x][y];
            G[x][y] = 255 - G[x][y];
            B[x][y] = 255 - B[x][y];
        }
    }
}
```

Purpose: Inverts the colors of the image.

Why: This creates a "negative" effect, which is a simple yet visually striking transformation.

ColorFilter Function

c

Copy

```
void ColorFilter(unsigned char R[WIDTH][HEIGHT], unsigned char G[WIDTH][HEIGHT],
unsigned char B[WIDTH][HEIGHT], int target_r, int target_g, int target_b, int threshold, int
replace_r, int replace_g, int replace_b) {
```

```
    int x, y;
    for (y = 0; y < HEIGHT; y++) {
        for (x = 0; x < WIDTH; x++) {
            if ((R[x][y] >= target_r - threshold && R[x][y] <= target_r + threshold) &&
                (G[x][y] >= target_g - threshold && G[x][y] <= target_g + threshold) &&
                (B[x][y] >= target_b - threshold && B[x][y] <= target_b + threshold)) {
                R[x][y] = replace_r;
                G[x][y] = replace_g;
                B[x][y] = replace_b;
            }
        }
    }
}
```

Purpose: Replaces pixels within a specific color range with a new color.

Why: This allows for selective color manipulation, useful for artistic effects or corrections.

Edge Function

c

Copy

```
void Edge(unsigned char R[WIDTH][HEIGHT], unsigned char G[WIDTH][HEIGHT], unsigned
char B[WIDTH][HEIGHT]) {
    int x, y;
    int temp_R, temp_G, temp_B;
    unsigned char new_R[WIDTH][HEIGHT], new_G[WIDTH][HEIGHT],
new_B[WIDTH][HEIGHT];
    ...
}
```

Purpose: Detects edges in the image using a simple convolution kernel.

Why: Edge detection is a fundamental operation in image processing, often used for feature extraction.

Shuffle Function

c

Copy

```
void Shuffle(unsigned char R[WIDTH][HEIGHT], unsigned char G[WIDTH][HEIGHT], unsigned
char B[WIDTH][HEIGHT]) {
    int x_separations[5] = {0, WIDTH / 4, WIDTH / 2, WIDTH * 3 / 4, WIDTH};
    int y_separations[5] = {0, HEIGHT / 4, HEIGHT / 2, HEIGHT * 3 / 4, HEIGHT};
    ...
}
```

Purpose: Rearranges blocks of the image to create a shuffled effect.

Why: This adds an artistic or puzzle-like effect to the image.

VFlip and HMirror Functions
c
Copy
```
void VFlip(unsigned char R[WIDTH][HEIGHT], unsigned char G[WIDTH][HEIGHT], unsigned
char B[WIDTH][HEIGHT]) {
    int x, y;
    unsigned char temp_R[WIDTH][HEIGHT], temp_G[WIDTH][HEIGHT],
temp_B[WIDTH][HEIGHT];
    ...
}

void HMirror(unsigned char R[WIDTH][HEIGHT], unsigned char G[WIDTH][HEIGHT], unsigned
char B[WIDTH][HEIGHT]) {
    int x, y;
    ...
}
```
Purpose: Flips the image vertically or mirrors it horizontally.

Why: These are basic geometric transformations that can alter the image's orientation.

Pixelate Function
c
Copy
```
void Pixelate(unsigned char R[WIDTH][HEIGHT], unsigned char G[WIDTH][HEIGHT], unsigned
char B[WIDTH][HEIGHT], int block_size) {
    int pixel_area = block_size * block_size;
    ...
}
```
Purpose: Reduces the image resolution by averaging pixel values in blocks.

Why: This creates a pixelated effect, often used for stylistic purposes or to obscure details.

7. AutoTest Function
c
Copy
```
void AutoTest(unsigned char R[WIDTH][HEIGHT], unsigned char G[WIDTH][HEIGHT], unsigned
char B[WIDTH][HEIGHT]) {
    char fname[SLEN] = "EngPlaza";
    char sname[SLEN];
    LoadImage(fname, R, G, B);
    Negative(R, G, B);
    strcpy(sname, "negative");
```

```
    SaveImage(sname, R, G, B);
    ...
}
```
Purpose: Tests all image processing functions automatically.

Why: This ensures that all functions work correctly and saves time during development.

8. Helper Functions

swap_blocks Function

c

Copy

```
void swap_blocks(int start_x1, int stop_x1, int start_y1, int stop_y1,
                 int start_x2, int stop_x2, int start_y2, int stop_y2,
                 unsigned char R[WIDTH][HEIGHT],
                 unsigned char G[WIDTH][HEIGHT],
                 unsigned char B[WIDTH][HEIGHT]) {
    ...
}
```
Purpose: Swaps two blocks of pixels in the image.

Why: This is used by the Shuffle function to rearrange image blocks.