Name: Rodrigo Ignacio Rojas Garcia
Course: ECE 4310
Lab #: 2

Optical Character Recognition

In this project the student was to implement a matched filter (normalized cross-correlation) to recognize the letters in an image of a text. The student was provided with an input image "parenthood.ppm", template image of desired character to be found "parenthood_e_template.ppm", and a ground truth file "parenthood_gt.txt" which was used to determine if the desired letter was recognized from the matched filter image.

The second laboratory was divided into five steps:

1. Read in the input image, template image, and ground truth file.
2. Calculate the matched-spatial filter (MSF) image
3. Normalize the MSF image into 8-bits
4. Loop through following steps for a range of different Thresholds (T):
    1. Threshold at T the normalized MSF image to create a binary image.
    2. Loop through the ground truth letter locations.
    3. Categorize and count the detected letters as FP ("detected" but the letter is not 'e'") and TP ("detected" and the letter is 'e')
    4. Output the total FP and TP for each T.

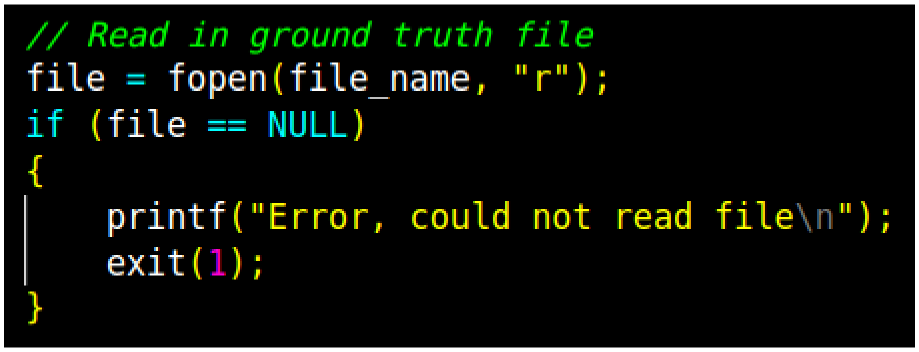
All results of each these were recorded. The MSF, both not normalized and normalized, were saved as gray-scale ppm images. Also, the total output of the total FP, TP, FN, FP TPR, FPR, and PPV were recorded with their corresponding threshold and documented into a CSV (Comma Separated Variable) file.

<u>STEP 1:</u>

Read in "parenthood.ppm" and "parenthood_e_template.ppm" code:

```c
unsigned char *read_in_image(int rows, int cols, char file_header[], FILE *image_file)
{
    // Variable Declaration Section
    unsigned char *image;

    image= (unsigned char *)calloc(rows * cols, sizeof(unsigned char));

    file_header[0] = fgetc(image_file);

    fread(image, sizeof(unsigned char), rows * cols, image_file);

    fclose(image_file);

    return image;
}
```

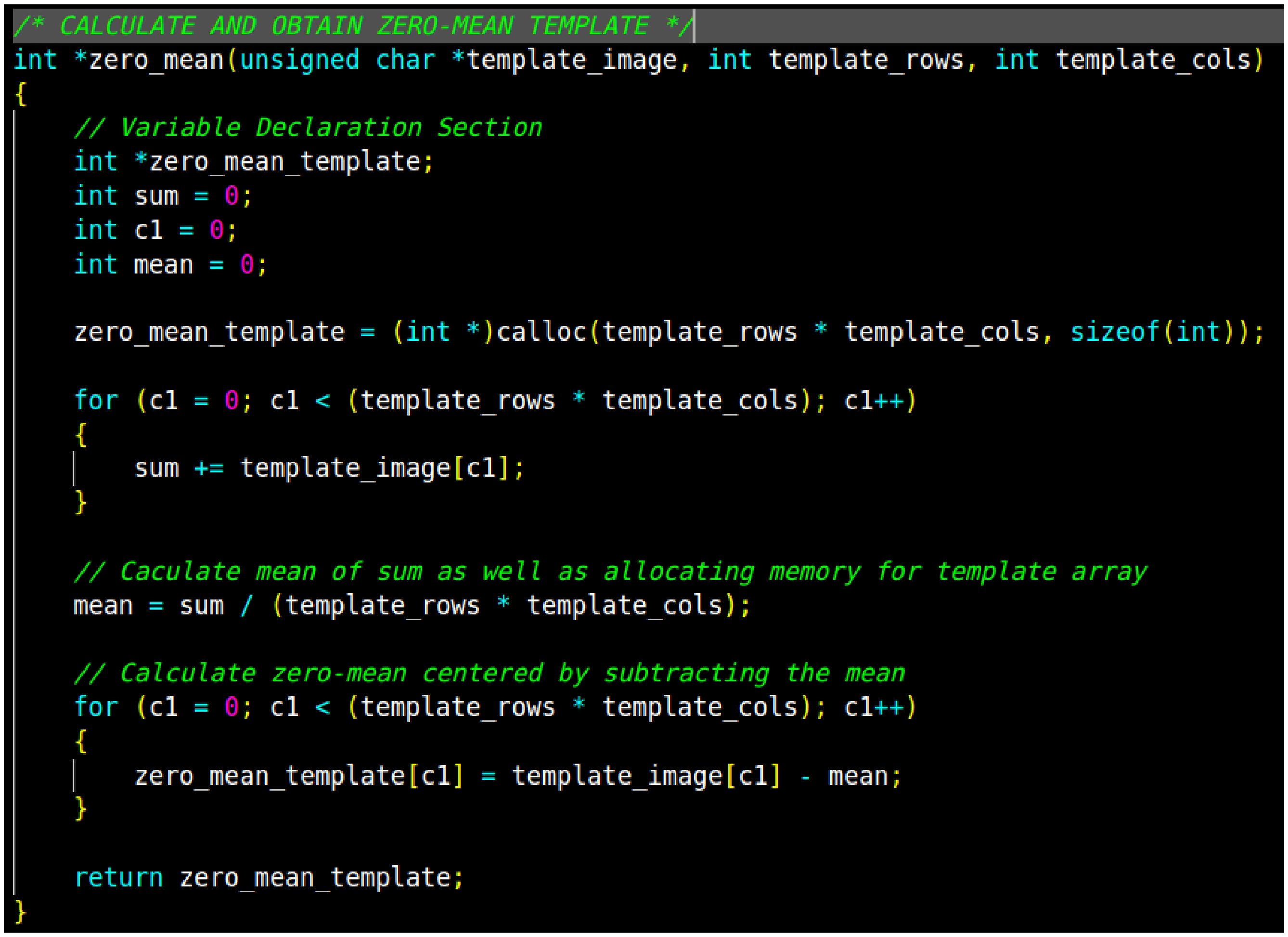
Read in "parenthood_gt.txt" code:



```c
// Read in ground truth file
file = fopen(file_name, "r");
if (file == NULL)
{
    printf("Error, could not read file\n");
    exit(1);
}
```

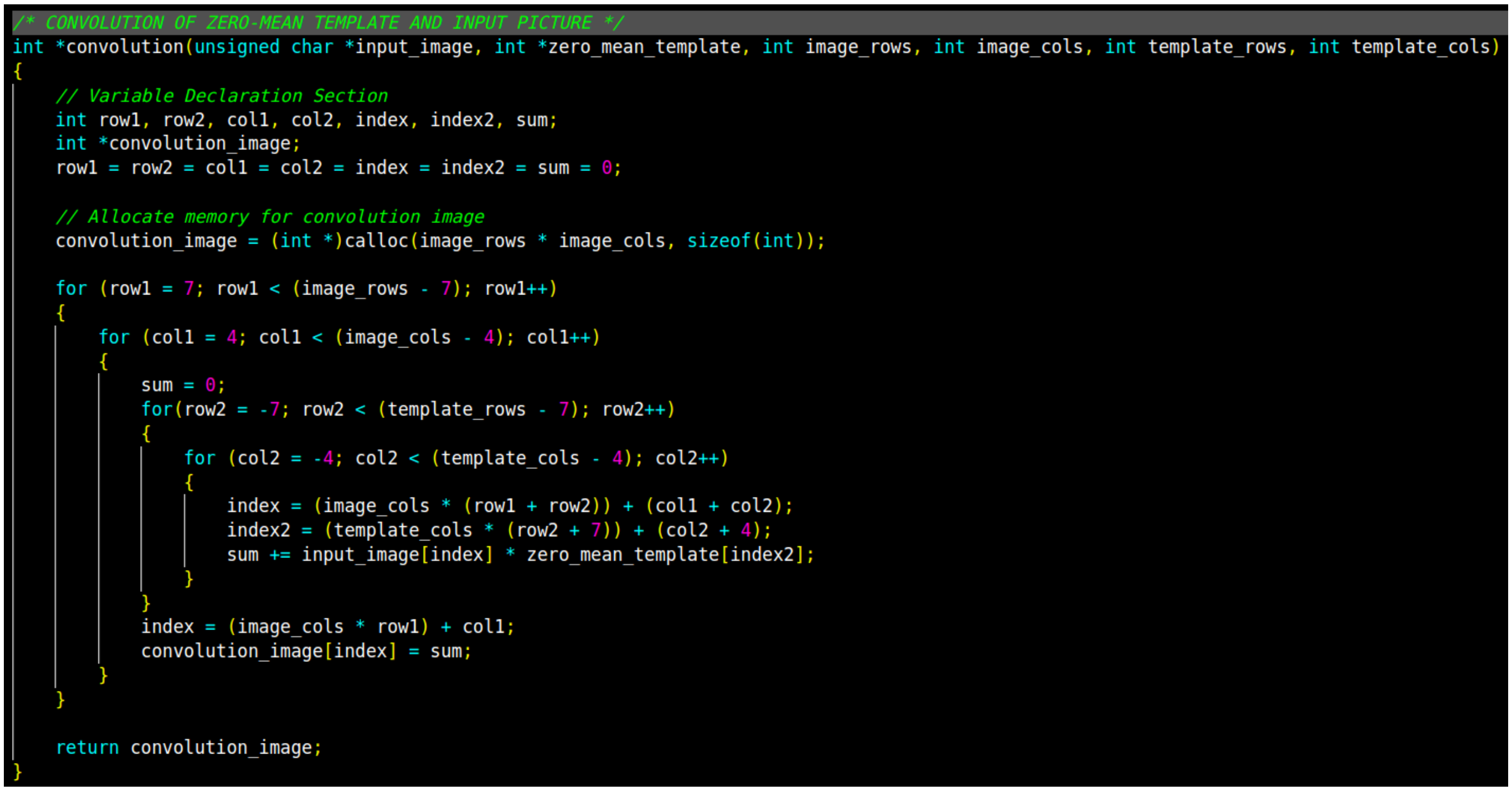
<u>STEP 2:</u>

Calculate Zero-Mean Template:



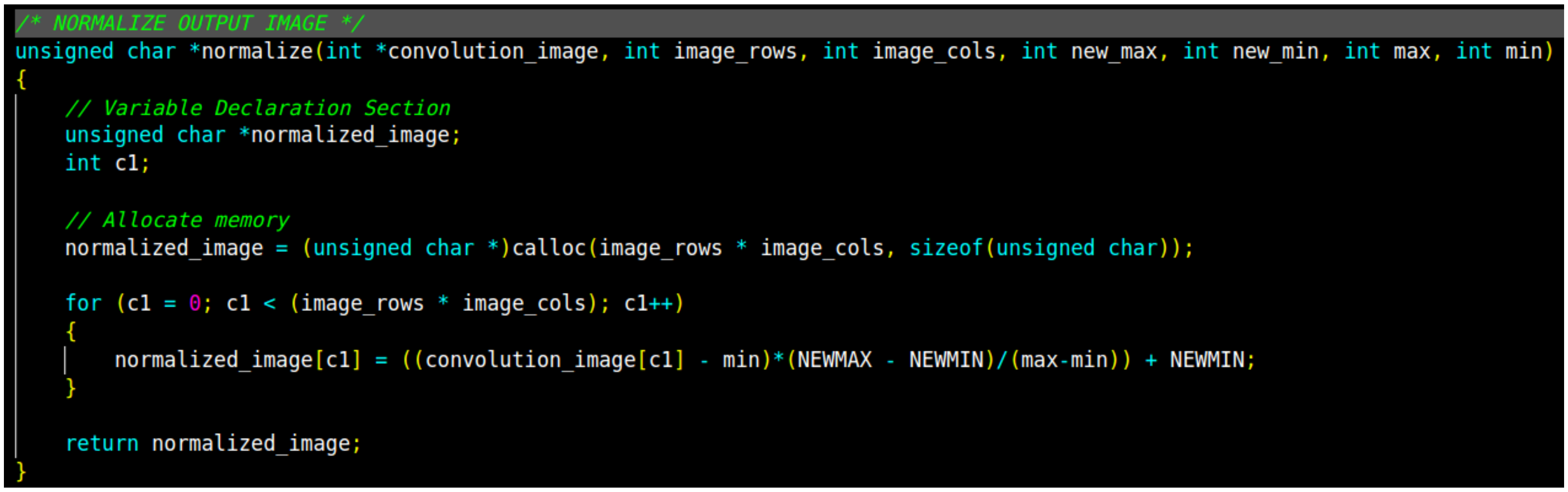
```c
/* CALCULATE AND OBTAIN ZERO-MEAN TEMPLATE */
int *zero_mean(unsigned char *template_image, int template_rows, int template_cols)
{
    // Variable Declaration Section
    int *zero_mean_template;
    int sum = 0;
    int c1 = 0;
    int mean = 0;

    zero_mean_template = (int *)calloc(template_rows * template_cols, sizeof(int));

    for (c1 = 0; c1 < (template_rows * template_cols); c1++)
    {
        sum += template_image[c1];
    }

    // Caculate mean of sum as well as allocating memory for template array
    mean = sum / (template_rows * template_cols);

    // Calculate zero-mean centered by subtracting the mean
    for (c1 = 0; c1 < (template_rows * template_cols); c1++)
    {
        zero_mean_template[c1] = template_image[c1] - mean;
    }

    return zero_mean_template;
}
```

Calculate MSF (not 8-bit) :



```c
/* CONVOLUTION OF ZERO-MEAN TEMPLATE AND INPUT PICTURE */
int *convolution(unsigned char *input_image, int *zero_mean_template, int image_rows, int image_cols, int template_rows, int template_cols)
{
    // Variable Declaration Section
    int row1, row2, col1, col2, index, index2, sum;
    int *convolution_image;
    row1 = row2 = col1 = col2 = index = index2 = sum = 0;

    // Allocate memory for convolution image
    convolution_image = (int *)calloc(image_rows * image_cols, sizeof(int));

    for (row1 = 7; row1 < (image_rows - 7); row1++)
    {
        for (col1 = 4; col1 < (image_cols - 4); col1++)
        {
            sum = 0;
            for(row2 = -7; row2 < (template_rows - 7); row2++)
            {
                for (col2 = -4; col2 < (template_cols - 4); col2++)
                {
                    index = (image_cols * (row1 + row2)) + (col1 + col2);
                    index2 = (template_cols * (row2 + 7)) + (col2 + 4);
                    sum += input_image[index] * zero_mean_template[index2];
                }
            }
            index = (image_cols * row1) + col1;
            convolution_image[index] = sum;
        }
    }

    return convolution_image;
}
```

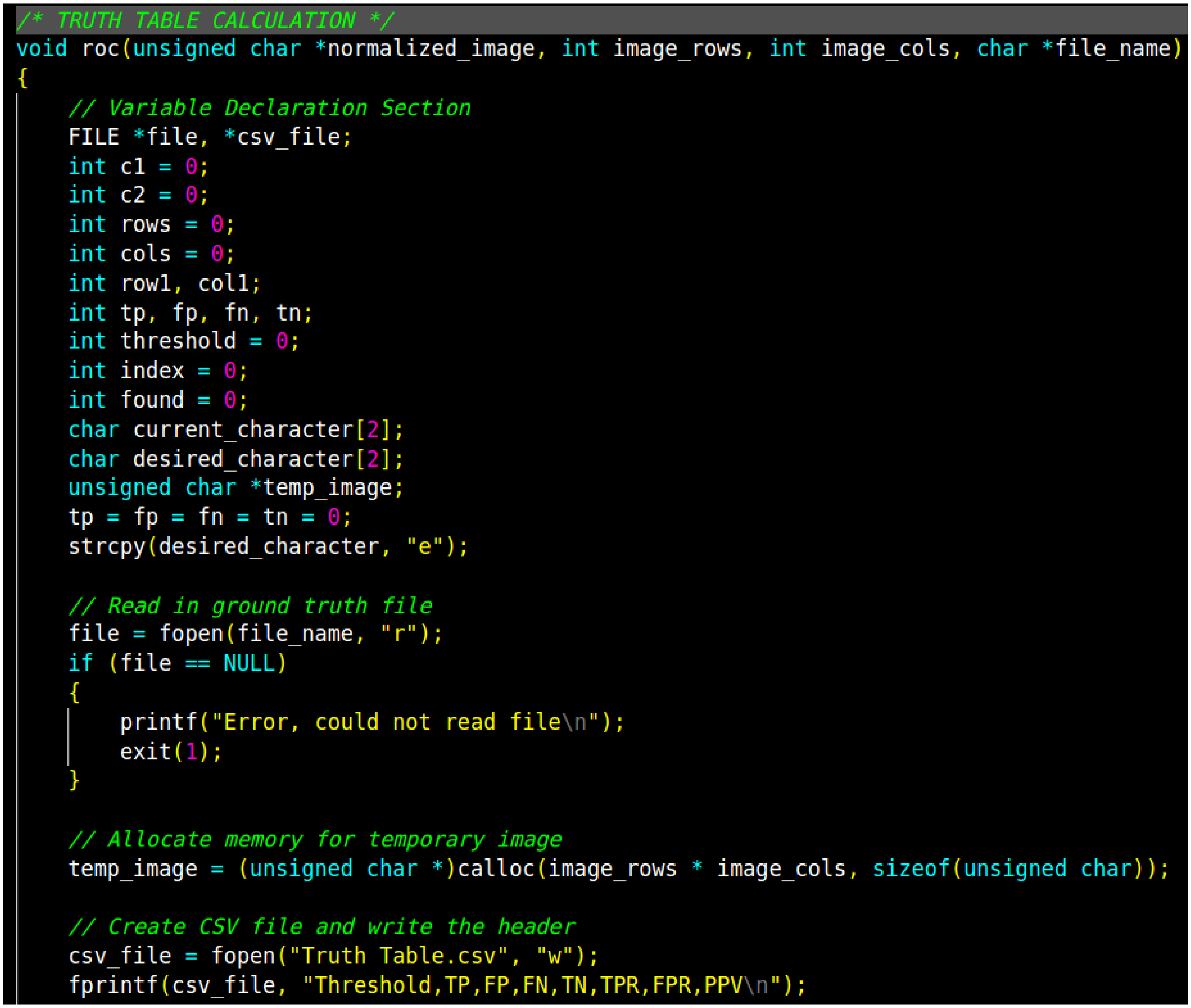
Normalize MSF:



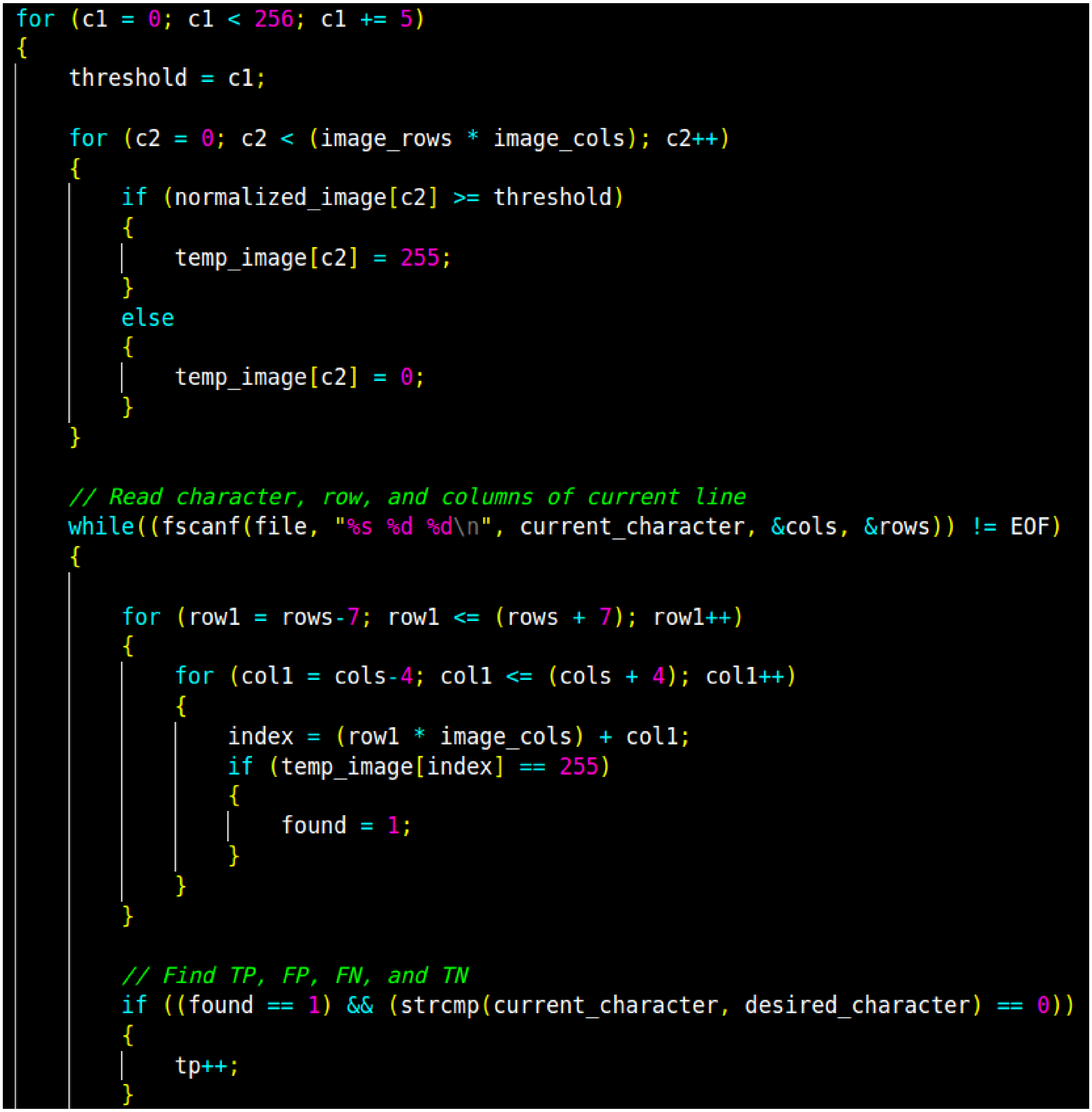
```c
/* NORMALIZE OUTPUT IMAGE */
unsigned char *normalize(int *convolution_image, int image_rows, int image_cols, int new_max, int new_min, int max, int min)
{
    // Variable Declaration Section
    unsigned char *normalized_image;
    int c1;

    // Allocate memory
    normalized_image = (unsigned char *)calloc(image_rows * image_cols, sizeof(unsigned char));

    for (c1 = 0; c1 < (image_rows * image_cols); c1++)
    {
        normalized_image[c1] = ((convolution_image[c1] - min)*(NEWMAX - NEWMIN)/(max-min)) + NEWMIN;
    }

    return normalized_image;
}
```

STEP 4:

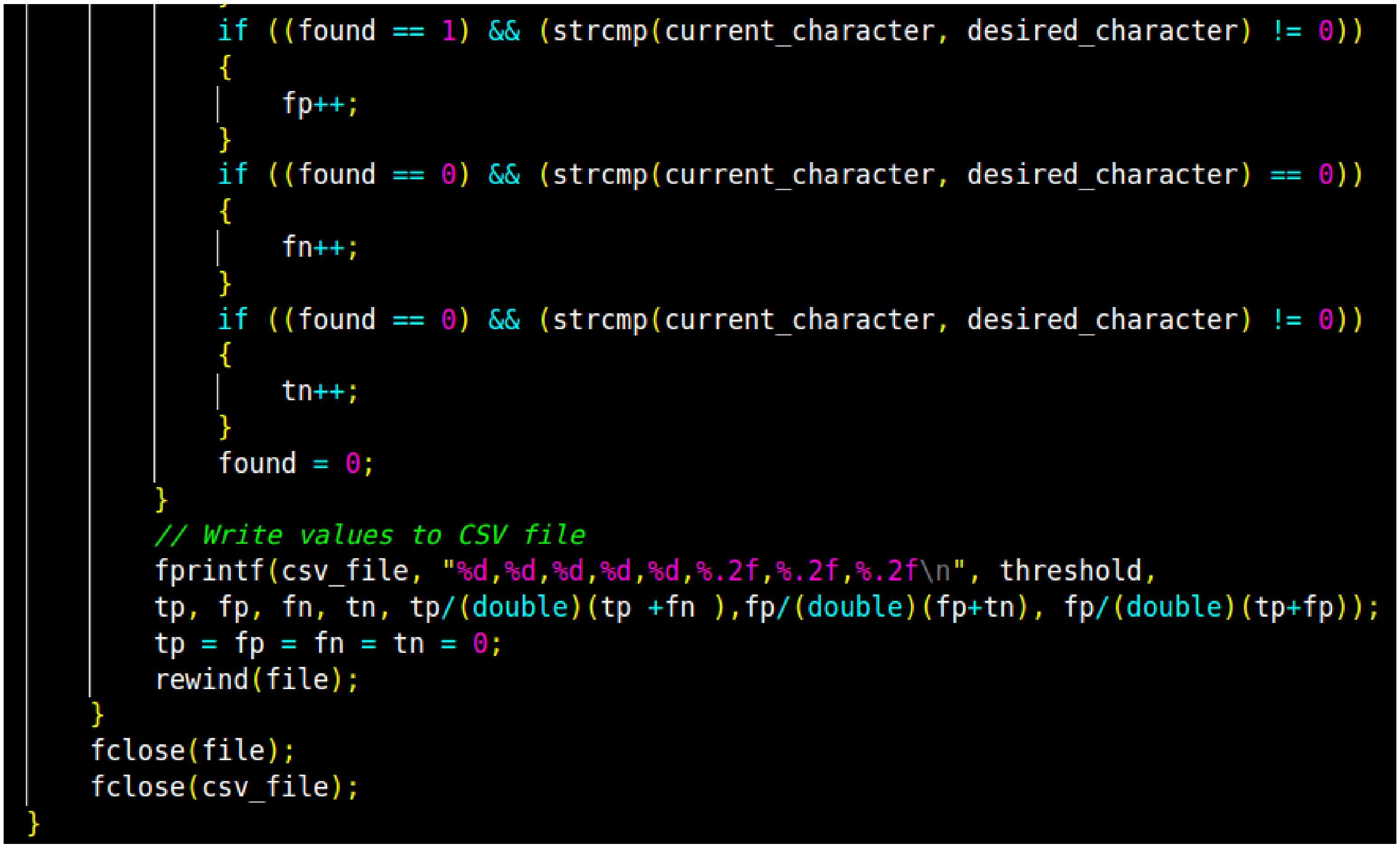
Loop through normalized MSF to calculate Truth Table with different Thresholds code:



```c
/* TRUTH TABLE CALCULATION */
void roc(unsigned char *normalized_image, int image_rows, int image_cols, char *file_name)
{
    // Variable Declaration Section
    FILE *file, *csv_file;
    int c1 = 0;
    int c2 = 0;
    int rows = 0;
    int cols = 0;
    int row1, col1;
    int tp, fp, fn, tn;
    int threshold = 0;
    int index = 0;
    int found = 0;
    char current_character[2];
    char desired_character[2];
    unsigned char *temp_image;
    tp = fp = fn = tn = 0;
    strcpy(desired_character, "e");

    // Read in ground truth file
    file = fopen(file_name, "r");
    if (file == NULL)
    {
        printf("Error, could not read file\n");
        exit(1);
    }

    // Allocate memory for temporary image
    temp_image = (unsigned char *)calloc(image_rows * image_cols, sizeof(unsigned char));

    // Create CSV file and write the header
    csv_file = fopen("Truth Table.csv", "w");
    fprintf(csv_file, "Threshold,TP,FP,FN,TN,TPR,FPR,PPV\n");
```



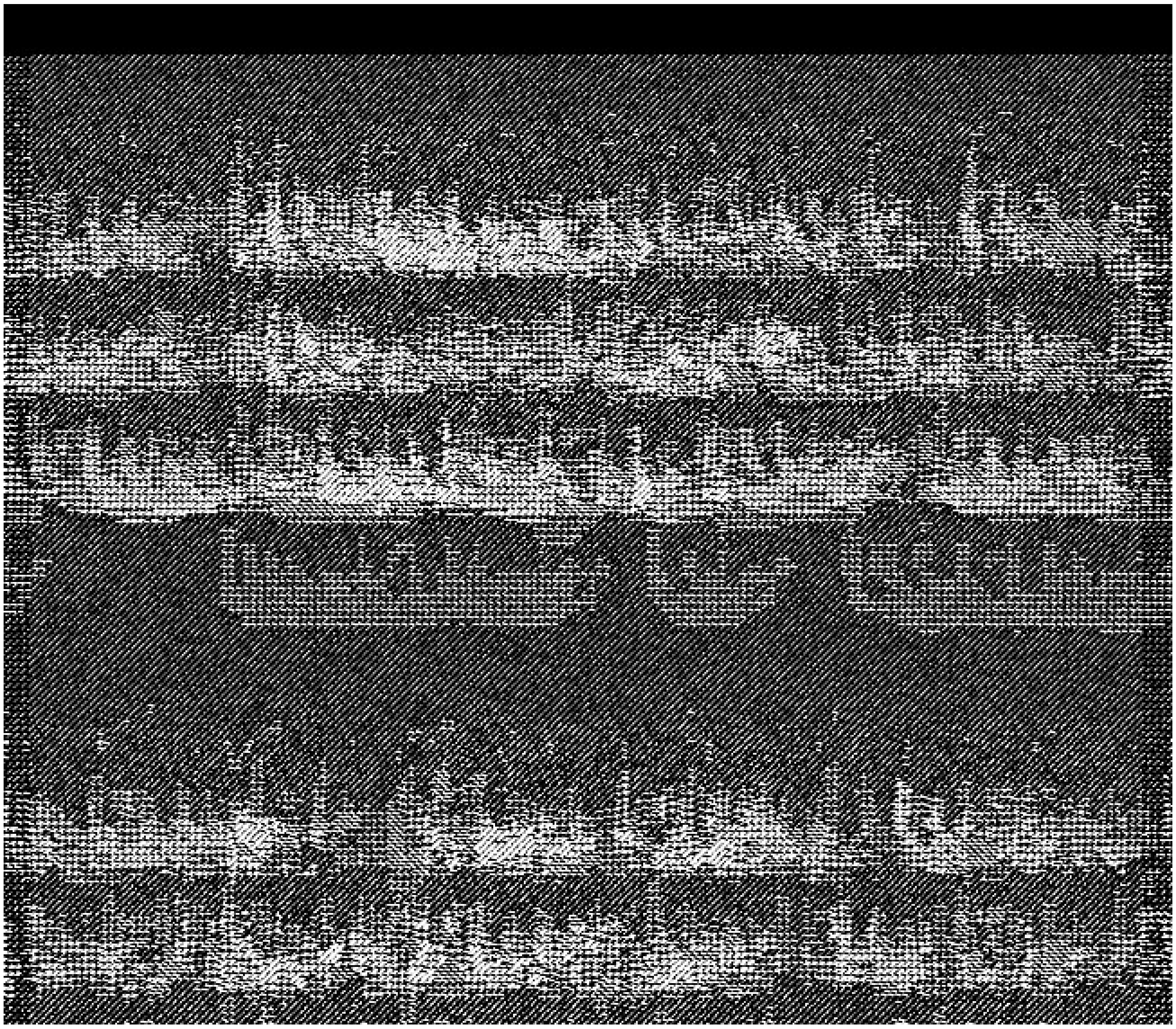
```
for (c1 = 0; c1 < 256; c1 += 5)
{
    threshold = c1;

    for (c2 = 0; c2 < (image_rows * image_cols); c2++)
    {
        if (normalized_image[c2] >= threshold)
        {
            temp_image[c2] = 255;
        }
        else
        {
            temp_image[c2] = 0;
        }
    }

    // Read character, row, and columns of current line
    while((fscanf(file, "%s %d %d\n", current_character, &cols, &rows)) != EOF)
    {

        for (row1 = rows-7; row1 <= (rows + 7); row1++)
        {
            for (col1 = cols-4; col1 <= (cols + 4); col1++)
            {
                index = (row1 * image_cols) + col1;
                if (temp_image[index] == 255)
                {
                    found = 1;
                }
            }
        }

        // Find TP, FP, FN, and TN
        if ((found == 1) && (strcmp(current_character, desired_character) == 0))
        {
            tp++;
        }
```



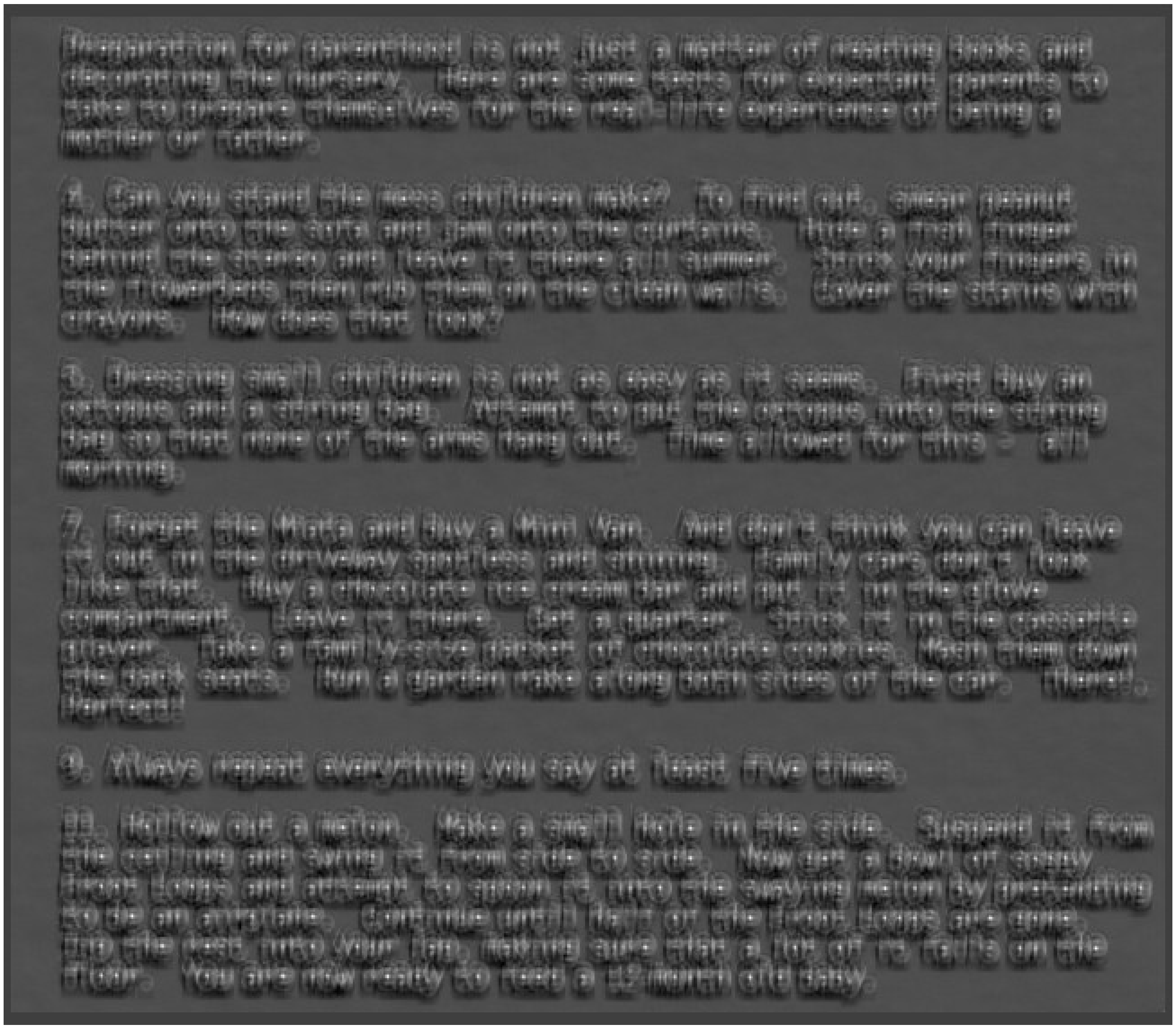
```
            if ((found == 1) && (strcmp(current_character, desired_character) != 0))
            {
                fp++;
            }
            if ((found == 0) && (strcmp(current_character, desired_character) == 0))
            {
                fn++;
            }
            if ((found == 0) && (strcmp(current_character, desired_character) != 0))
            {
                tn++;
            }
            found = 0;
        }
        // Write values to CSV file
        fprintf(csv_file, "%d,%d,%d,%d,%d,%.2f,%.2f,%.2f\n", threshold,
        tp, fp, fn, tn, tp/(double)(tp +fn ),fp/(double)(fp+tn), fp/(double)(tp+fp));
        tp = fp = fn = tn = 0;
        rewind(file);
    }
    fclose(file);
    fclose(csv_file);
}
```

RESULTS:

MSF Image (not 8-bit)



Normalized MSF Image (8-bit):



Preparation for parenthood is not just a matter of reading books and decorating the nursery. Here are some tests for expectant parents to take to prepare themselves for the real-life experience of being a mother or father.

1. Can you stand the mess children make? To find out, smear peanut butter onto the sofa and jam onto the curtains. Hide a fish finger behind the stereo and leave it there all summer. Stick your fingers in the flowerbeds then rub them on the clean walls. Cover the stains with crayons. How does that look?

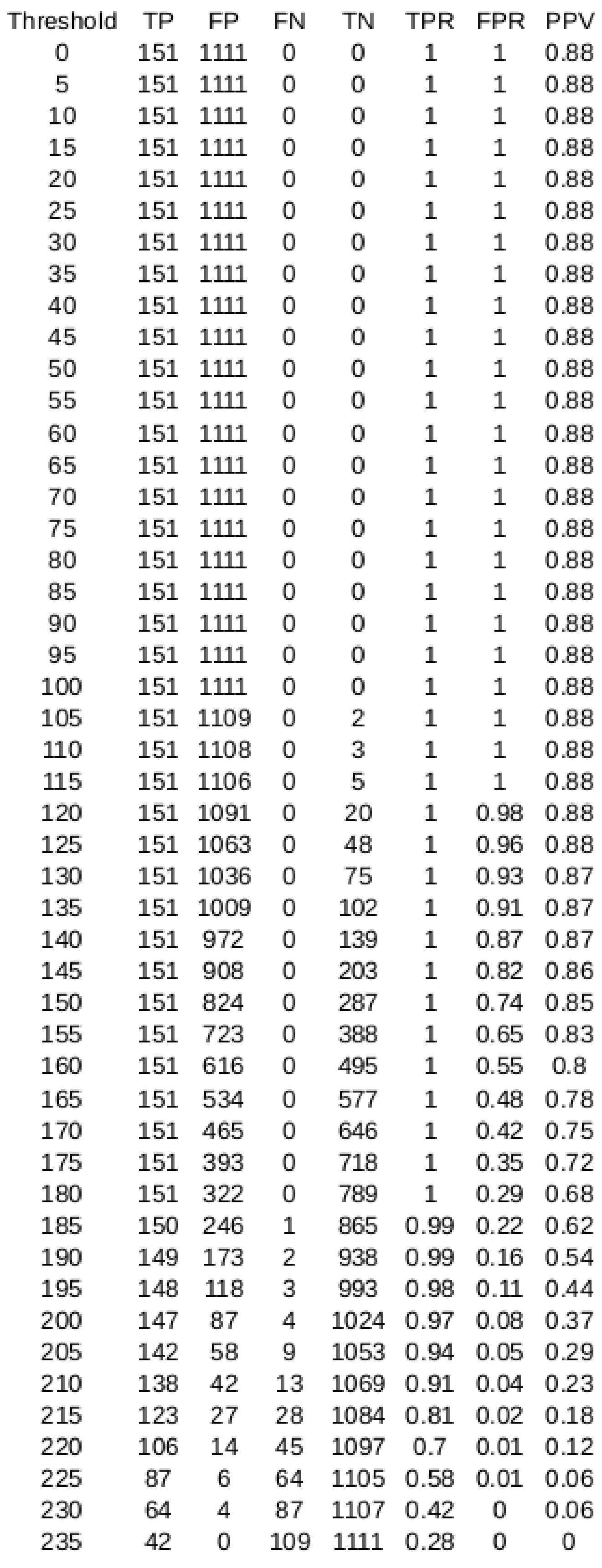
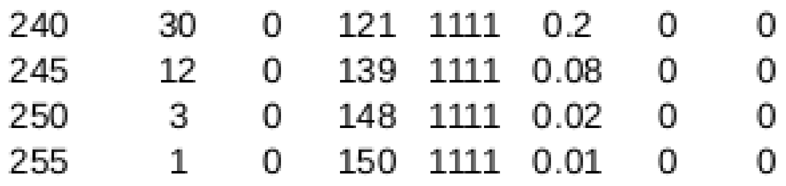
3. Dressing small children is not as easy as it seems. First buy an octopus and a string bag. Attempt to put the octopus into the string bag so that none of the arms hang out. Time allowed for this - all morning.

7. Forget the Miata and buy a Mini Van. And don't think you can leave it out in the driveway spotless and shining. Family cars don't look like that. Buy a chocolate ice cream bar and put it in the glove compartment. Leave it there. Get a quarter. Stick it in the cassette player. Take a family-size packet of chocolate cookies. Mash them down the back seats. Run a garden rake along both sides of the car. There! Perfect!

9. Always repeat everything you say at least five times.

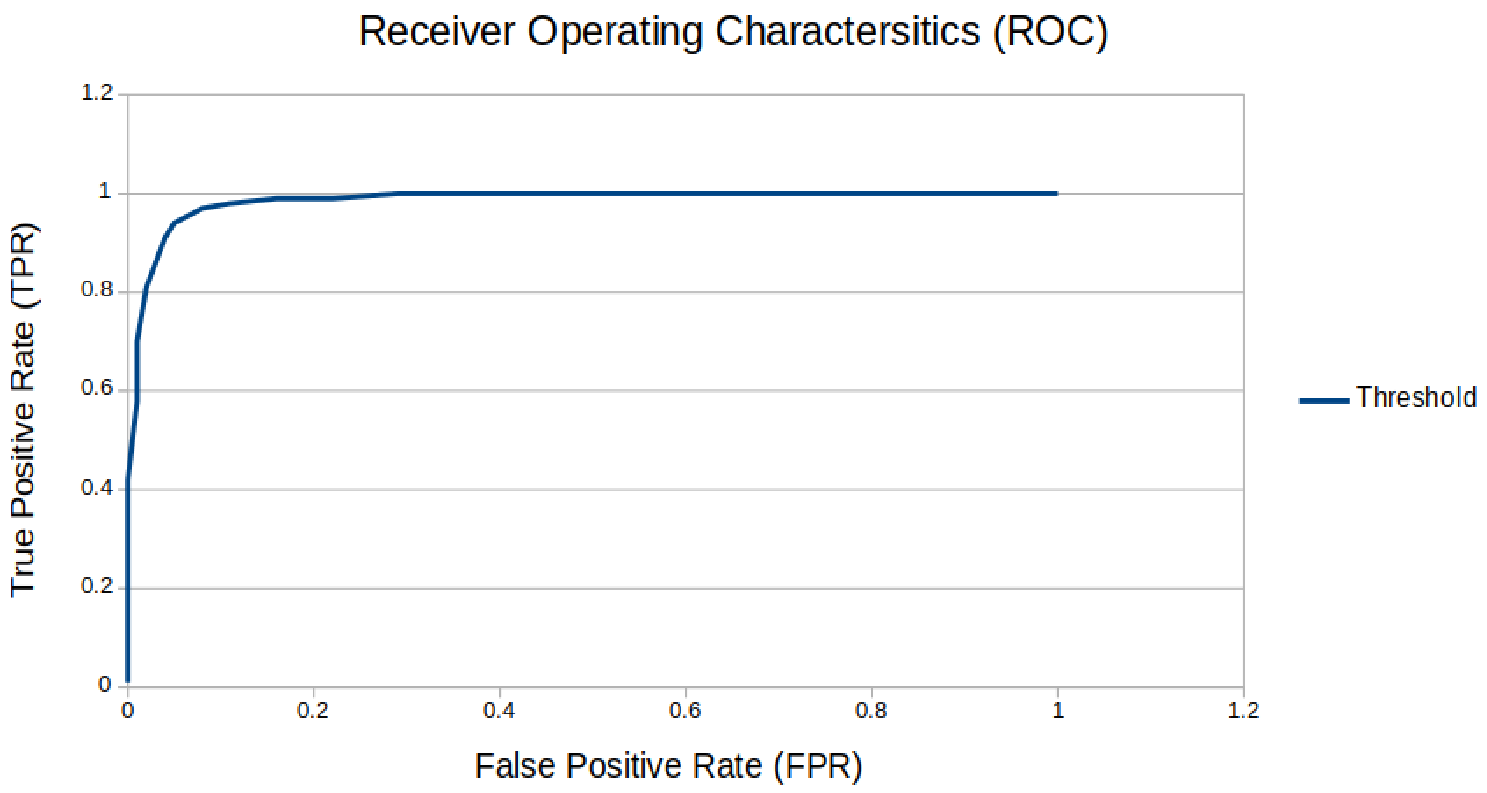
11. Hollow out a melon. Make a small hole in the side. Suspend it from the ceiling and swing it from side to side. Now get a bowl of soggy Froot Loops and attempt to spoon it into the swaying melon by pretending to be an airplane. Continue until half of the Froot Loops are gone. Tip the rest into your lap, making sure that a lot of it falls on the floor. You are now ready to feed a 12-month old baby.

Truth Table with different Thresholds:

Threshold	TP	FP	FN	TN	TPR	FPR	PPV
0	151	1111	0	0	1	1	0.88
5	151	1111	0	0	1	1	0.88
10	151	1111	0	0	1	1	0.88
15	151	1111	0	0	1	1	0.88
20	151	1111	0	0	1	1	0.88
25	151	1111	0	0	1	1	0.88
30	151	1111	0	0	1	1	0.88
35	151	1111	0	0	1	1	0.88
40	151	1111	0	0	1	1	0.88
45	151	1111	0	0	1	1	0.88
50	151	1111	0	0	1	1	0.88
55	151	1111	0	0	1	1	0.88
60	151	1111	0	0	1	1	0.88
65	151	1111	0	0	1	1	0.88
70	151	1111	0	0	1	1	0.88
75	151	1111	0	0	1	1	0.88
80	151	1111	0	0	1	1	0.88
85	151	1111	0	0	1	1	0.88
90	151	1111	0	0	1	1	0.88
95	151	1111	0	0	1	1	0.88
100	151	1111	0	0	1	1	0.88
105	151	1109	0	2	1	1	0.88
110	151	1108	0	3	1	1	0.88
115	151	1106	0	5	1	1	0.88
120	151	1091	0	20	1	0.98	0.88
125	151	1063	0	48	1	0.96	0.88
130	151	1036	0	75	1	0.93	0.87
135	151	1009	0	102	1	0.91	0.87
140	151	972	0	139	1	0.87	0.87
145	151	908	0	203	1	0.82	0.86
150	151	824	0	287	1	0.74	0.85
155	151	723	0	388	1	0.65	0.83
160	151	616	0	495	1	0.55	0.8
165	151	534	0	577	1	0.48	0.78
170	151	465	0	646	1	0.42	0.75
175	151	393	0	718	1	0.35	0.72
180	151	322	0	789	1	0.29	0.68
185	150	246	1	865	0.99	0.22	0.62
190	149	173	2	938	0.99	0.16	0.54
195	148	118	3	993	0.98	0.11	0.44
200	147	87	4	1024	0.97	0.08	0.37
205	142	58	9	1053	0.94	0.05	0.29
210	138	42	13	1069	0.91	0.04	0.23
215	123	27	28	1084	0.81	0.02	0.18
220	106	14	45	1097	0.7	0.01	0.12
225	87	6	64	1105	0.58	0.01	0.06
230	64	4	87	1107	0.42	0	0.06
235	42	0	109	1111	0.28	0	0
240	30	0	121	1111	0.2	0	0
245	12	0	139	1111	0.08	0	0
250	3	0	148	1111	0.02	0	0
255	1	0	150	1111	0.01	0	0

Receiver Operating Characteristics (ROC):



Based on the ROC graph, the best threshold in which the found/not found letter ‘e’ ratio is at threshold 205. The following image shows the ‘e’ found at threshold 205: