Name: Rodrigo Ignacio Rojas Garcia

Course: ECE 4310

Lab #3

Letters

In this project the student was to implement thinning, branch point, and endpoint detection in order to recognize 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”, ground truth file “ground\_truth.txt”, and was required to use Match-Spatial Filter (MSF) image from laboratory 2 “msf\_e.ppm”. All of these images were used to determine if the desired letter was recognized from the MSF.

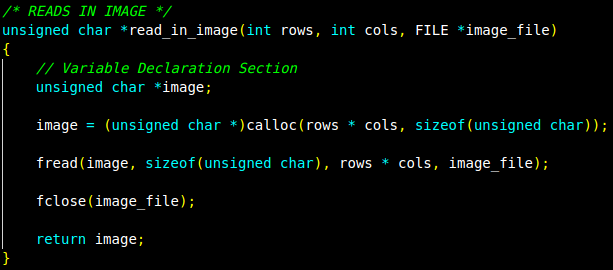
The third laboratory was divided into multiple steps:

1. Read in input image, template image, MSF image, and ground truth text file.
2. Threshold input image at a value of 128 to create a binary image.
3. Thin the thresholded image down to single-pixel wide components.
4. Determine end point and branch points and store it in a binary image.
5. Loop through the following steps for a range of T (Threshold):
   1. Threshold MSF image into a binary image based on T.
   2. Loop through ground truth letter locations.
   3. Determine if the letter is an “e” or not “e” based on T and if and only if the found “e” has exactly one endpoint and one branch point, if not letter is considered as not an “e”. If letter considered an “e”, add to value TP (“detected and letter is “e”), if not add to value FP (“detected” but the letter is not “e”).

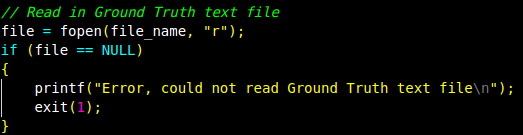
All results of each of these steps were recorded. All images were saved as gray-scale ppm images. Also, the total output of the total FP, TP, FN, FP, TPR, FPR, and PPV were recorded with their corresponding threshold and documented into a CSV (Comma Separated Variable) file.

**STEP 1:**

*Read in “parenthood.ppm”, “parenthood\_e\_template.ppm”, and “msf\_e.ppm” PPM files*

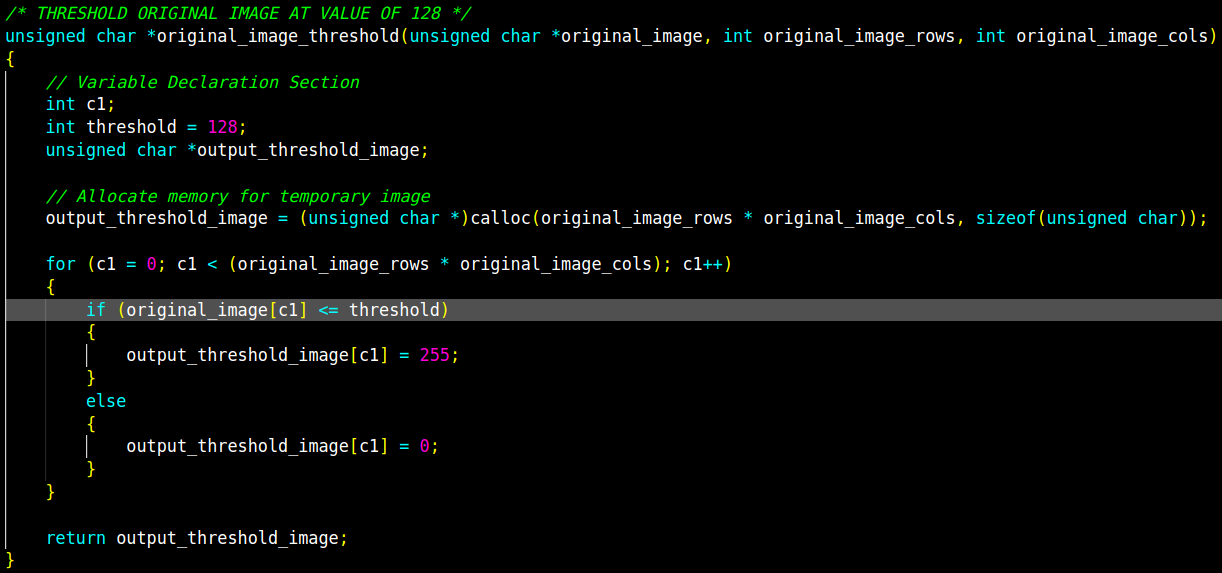


*Read in “ground\_truth.txt” text file*



***STEP 2***

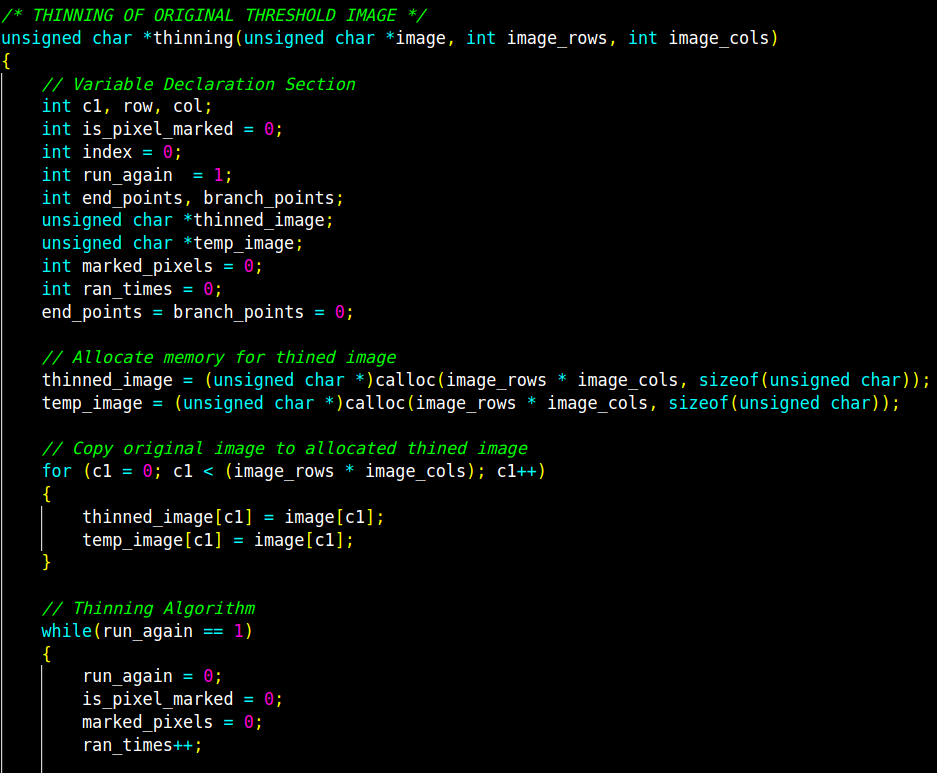
*Threshold “parenthood.ppm” PPM file*

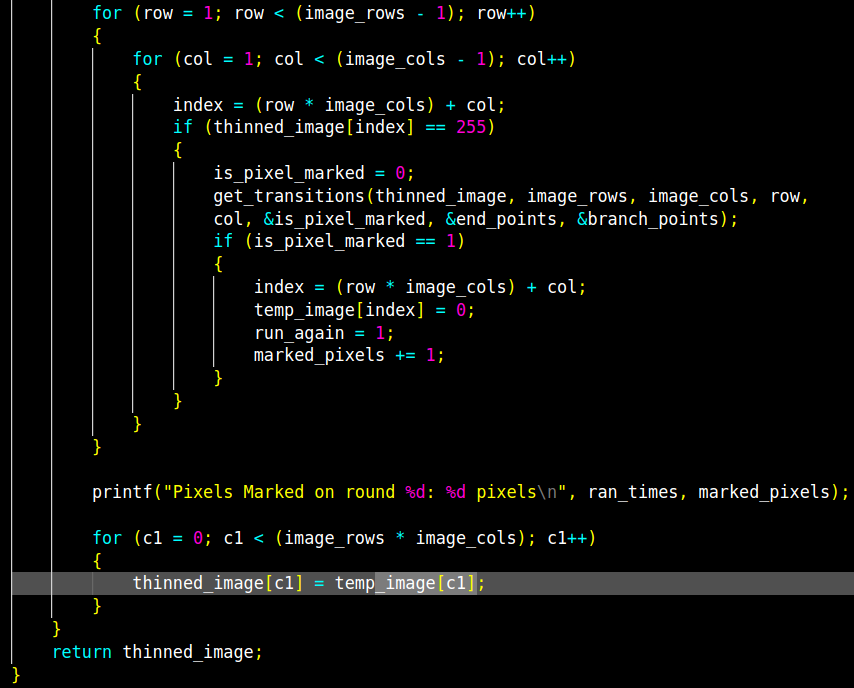


***STEP 3***

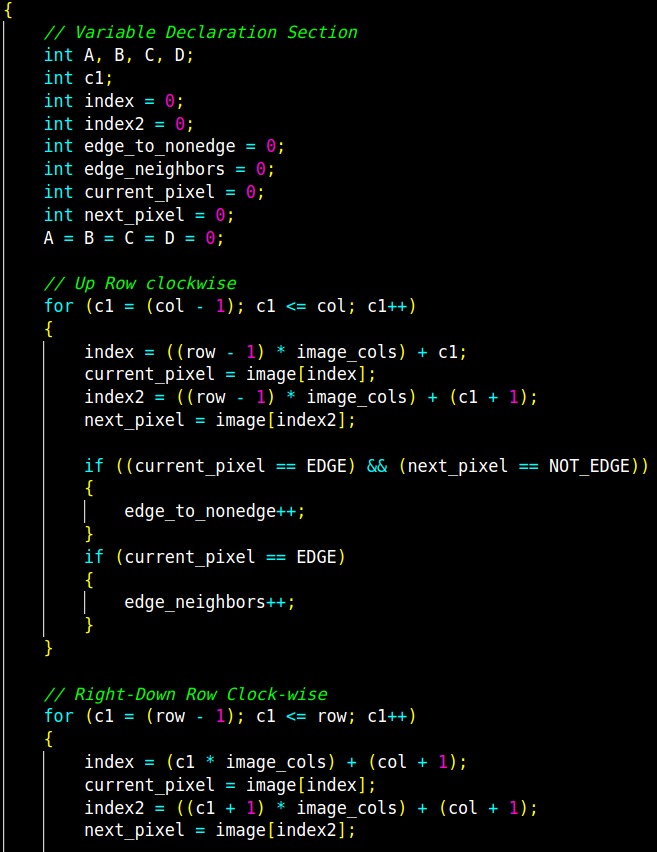
*Thin thresholded image down to pixel-wide components*

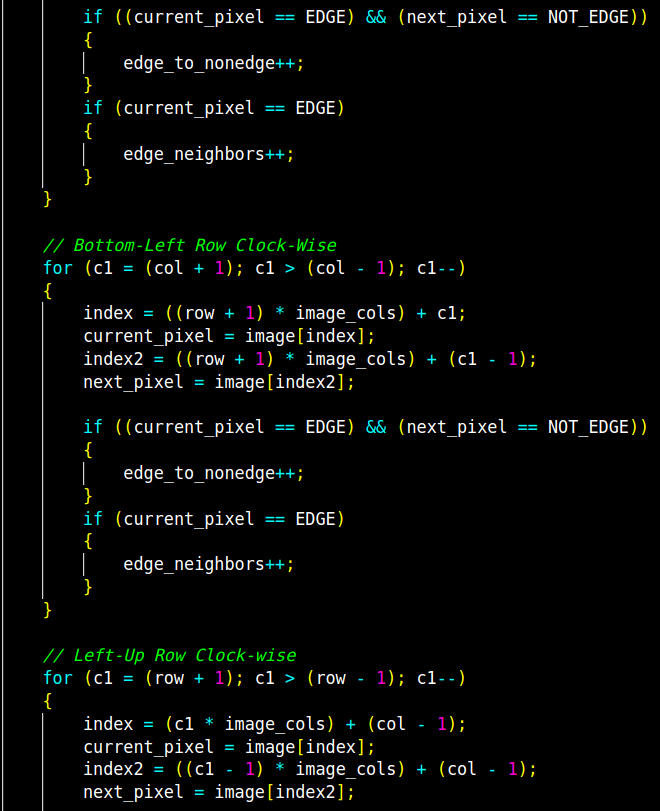
* *Function “thinning()” calls function “get\_transitions()” which determine if the pixel needs to be erased. The “get\_transitions()” function also calculates both end points and branch points on the thresholded image, but in this case they are not used.*

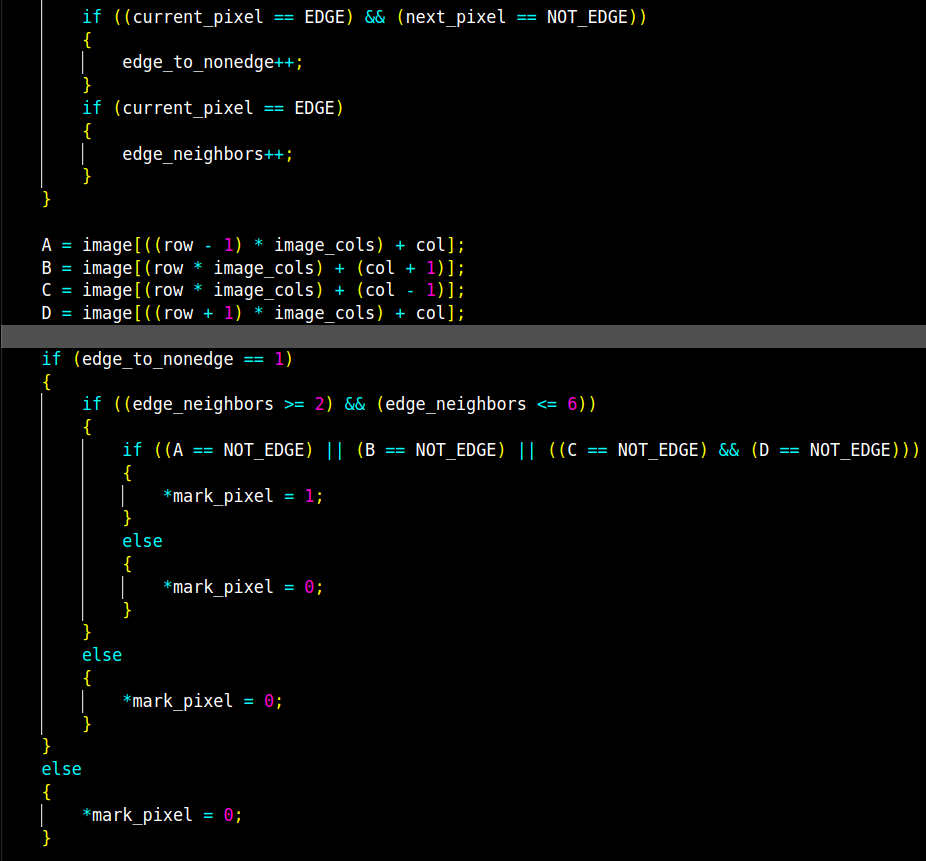


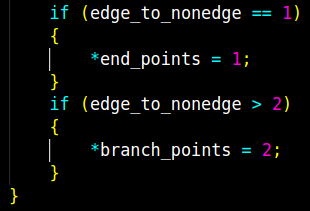










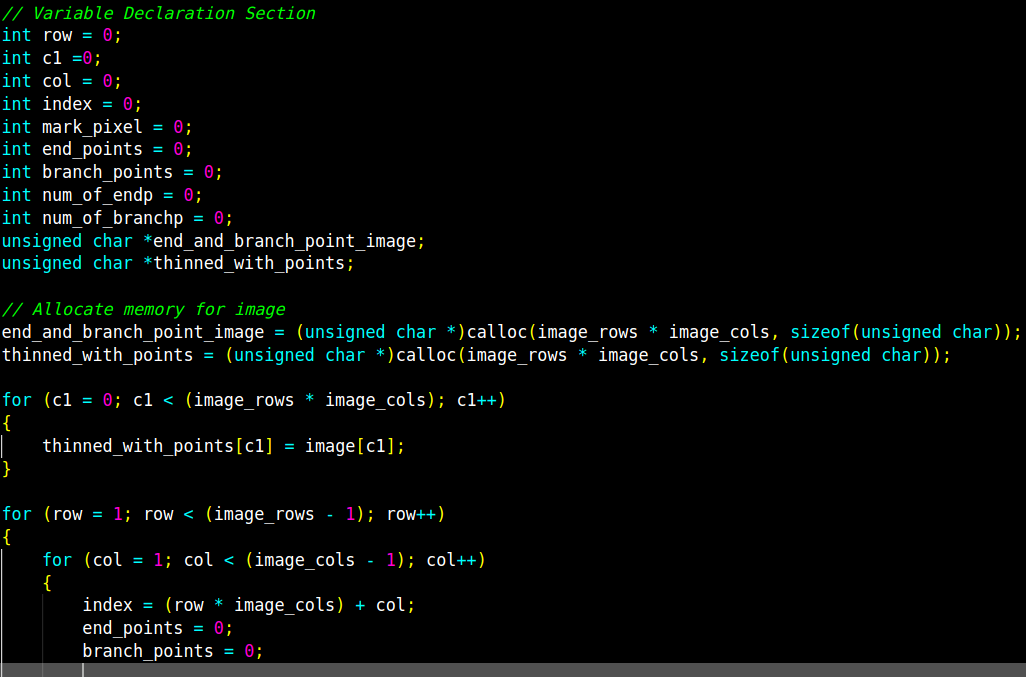


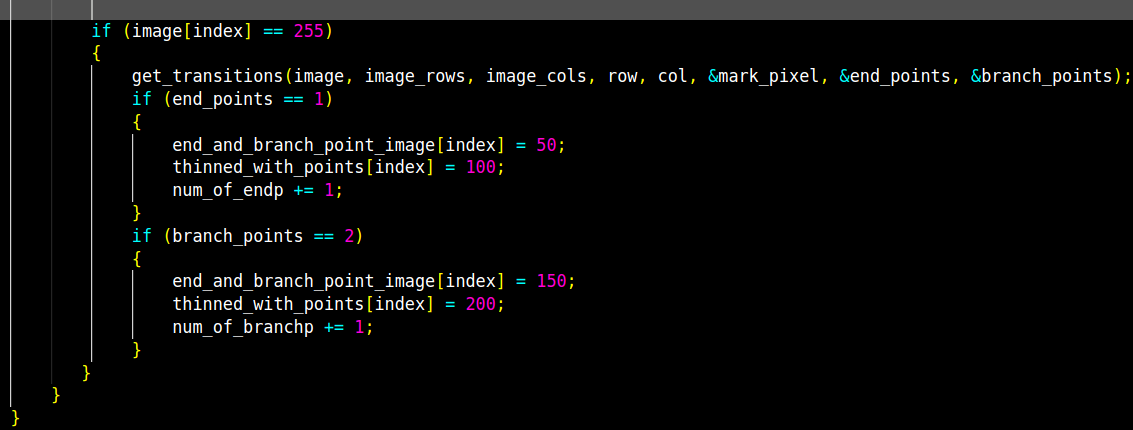
**STEP 4**

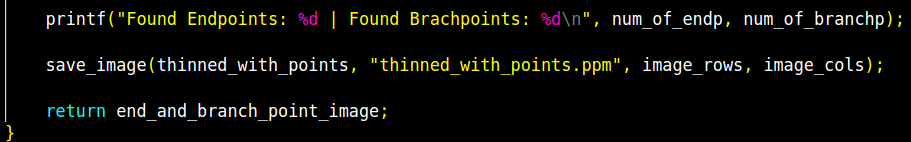
*Determine end points and branch points and store it on a binary image*

* *Function “get\_end\_and\_branch\_points()” also calls function “get\_transitions()” which will return both end points and branch points and values will be used. In this case, the “marked\_pixe()” variable will not be used.*
* *Function “get\_end\_and\_branch\_points()” will also create a binary image which will mark end points and branch points in the thinned image for report purposes.*



**

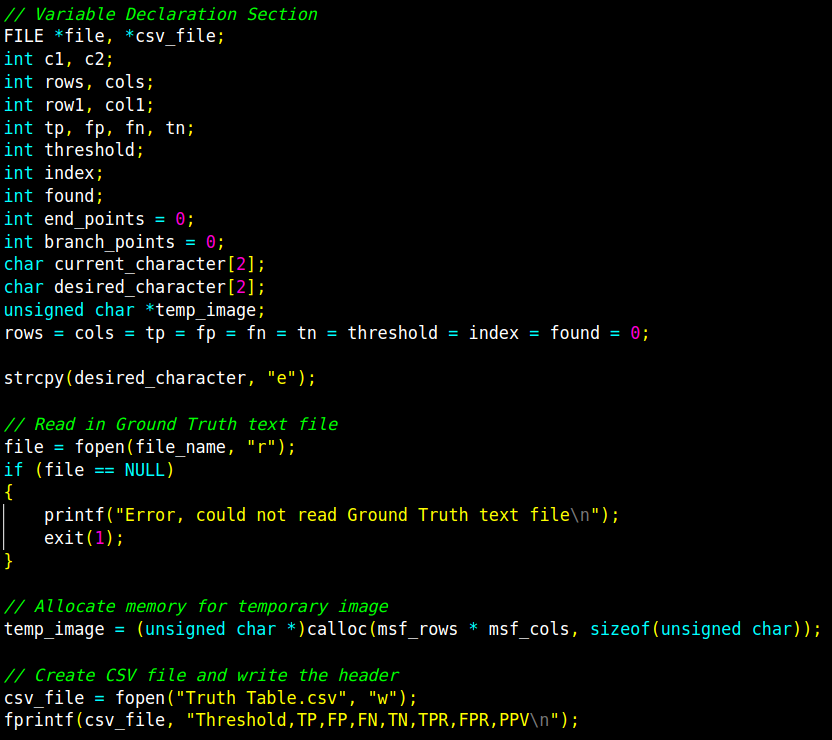
**

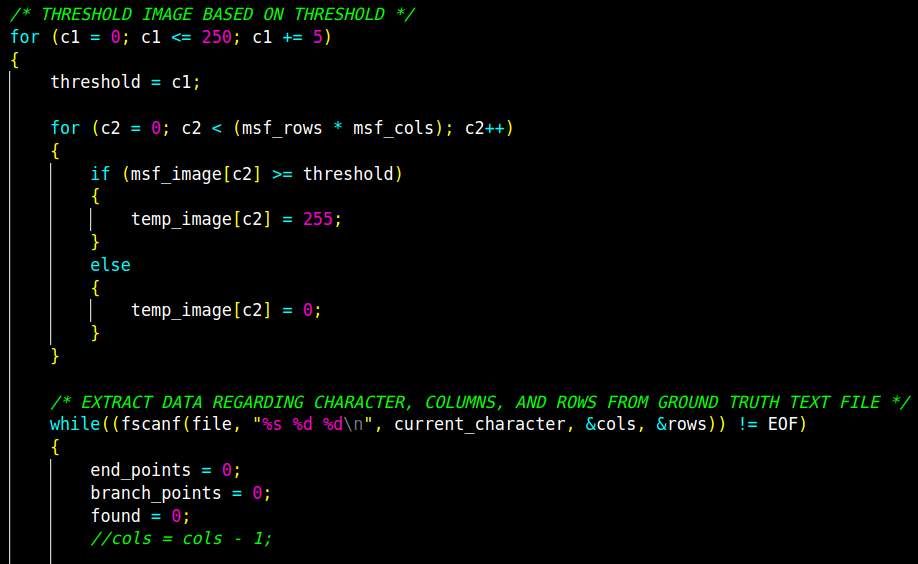
**

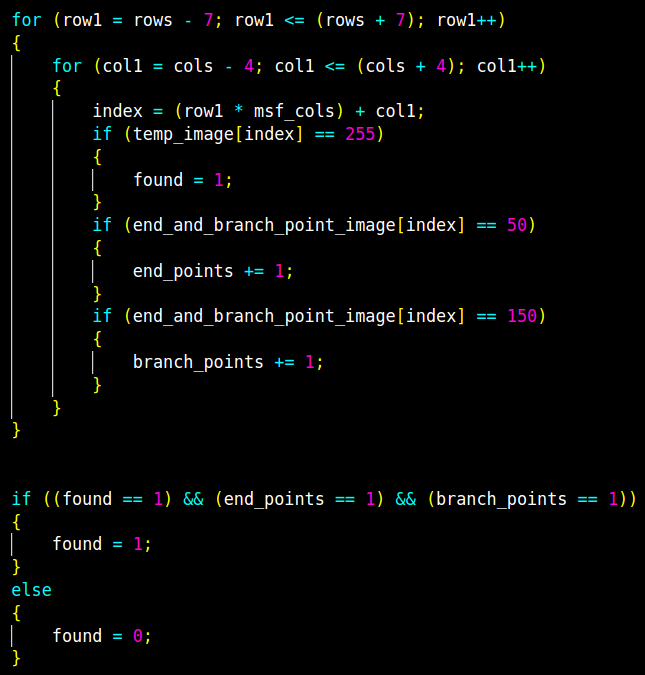
**STEP 5**

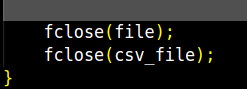
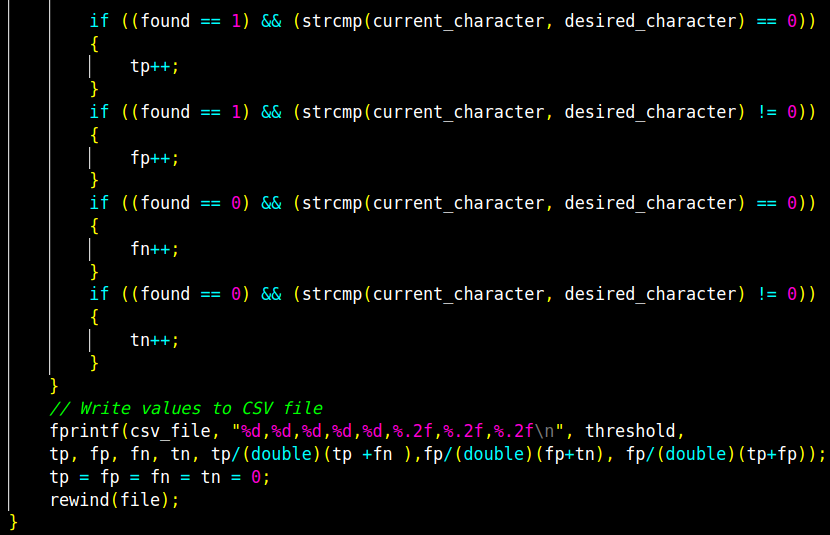
Threshold the MSF image with different T values, determine if letter is an “e” based on location given the ground truth text file, check if “e” is found with constraint of only one branch point and one end point, and finally save all TP, FP, TF, TN, FPR, TPN, and PPV to a CSV file.





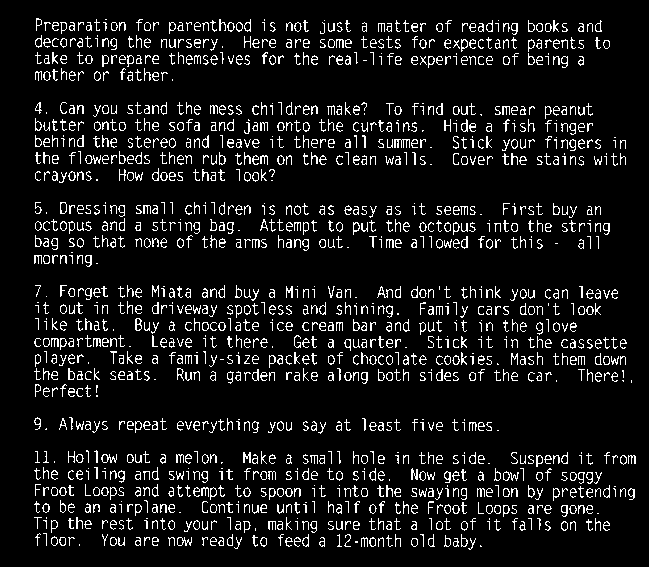




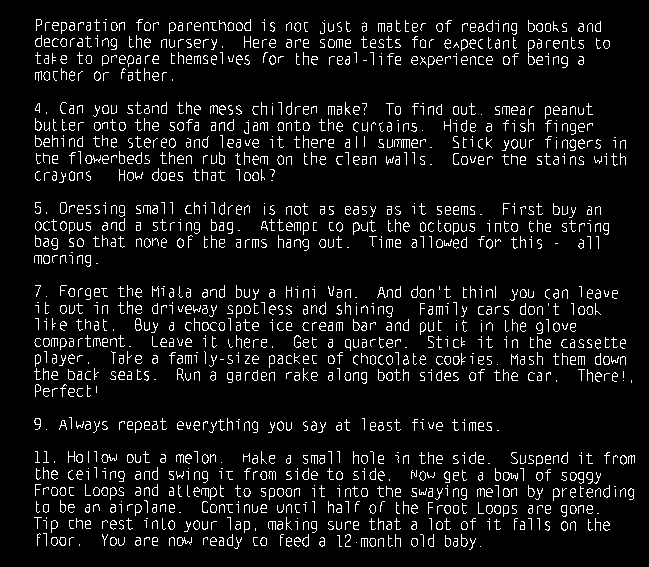


**RESULTS**

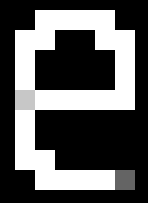
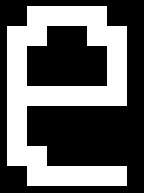
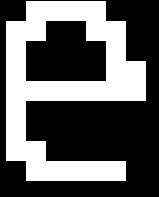
*Input mage with Threshold at value 128:*

**

*Thinned Image:*

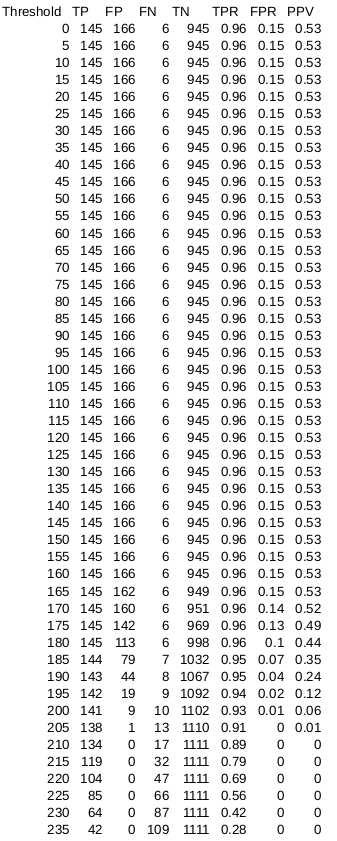
**

*Thinned image with endpoints and branch points example:*

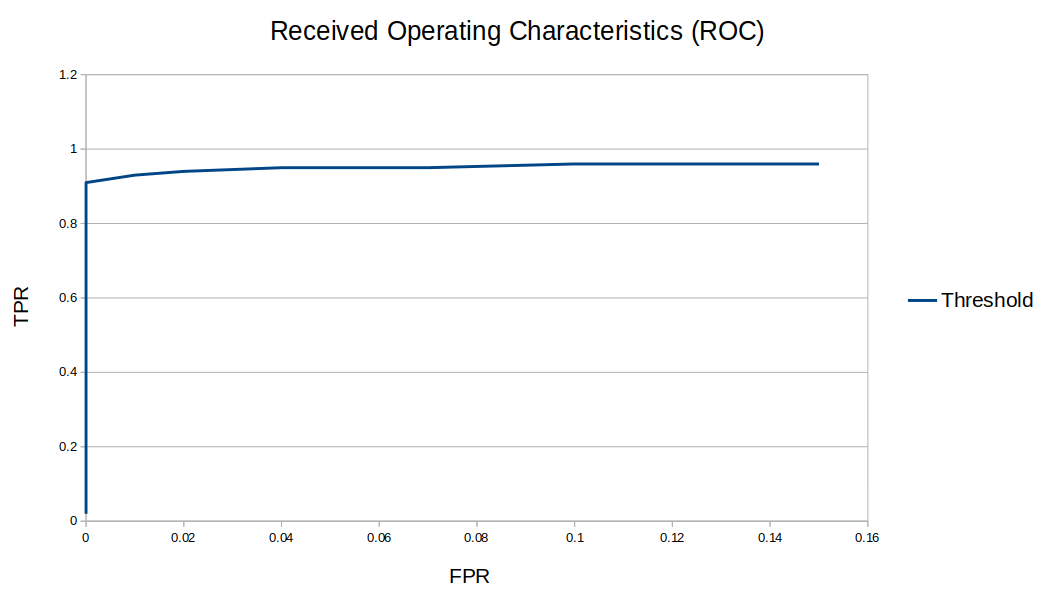
**

*Original “e” → 128 Threshold “e” → Thinned “e” → Thinned “e” with end point and branch point*

*Truth Table:*

**

*Received Operating Characteristics (ROC) :*

 Based on the ROC graph, the best thresholds in which the found/not found for letter “e” is at 200 with TP, FP, TPR, and FPR values of 141, 9, 0.93 and 0.01 correspondingly.