Problem1

My algorithm:
1) binarize both Sample1.raw and TrainingSet.raw with thresholds 30 and 50 respectively.
2) do dilation on both of the binarized images
3) extract components
   A. for Sample1, using connected component labeling to extract the components
   B. for TrainingSet
      1. splitting into 5 x 14 = 70 regions
      2. making the bounding box tighter
4) computing feature vectors for TrainingSet's components
   A. the function computeFeatureVector() will help compute the feature vector for each component
   B. the FeatureVector structure has some essential methods
      1. getLength()
      2. getSimilarity(), getSimilarity2(): getSimilarity() utilizes Euclidean distance, while getSimilarity2() utilizes cosine similarity.
5) iterate through every component in Sample1 and determine the best match
   A. first compute feature vector for the component in Sample1
   B. with getSimilarity(), we can determine the best match for this component.

As for the content of each FeatureVector, there are basically 11 elements. First, the height and width ratio are two of the features. (height_ratio = height / (height + width) and width_ratio = width / (height + width).) And each component is equally divided into 3 x 3 = 9 regions. The ratio of pixel count in foreground versus the pixel count in background is used as one of the features. Therefore, there are 2 + 9 = 11 features in each FeatureVector.

The result of my algorithm is: Y B i P R y R 7 T 3 H c T R B C P v T H 5 $,
where the ground truth is: Y 9 4 P 9 V 3 7 T 3 H C T 3 B C P V T H 5 B.
And my program has reached a 68.18% accuracy.

Problem 2

For (a), I've implemented the "skeletonize" morphology. This mathematical morphology can be achieved through some simple functions, such as UNION, DIFFERENCE, DILATION and EROSION. The formula for "skeletonize" is proposed by Mr. Lantuéjoul. And it is as follows:

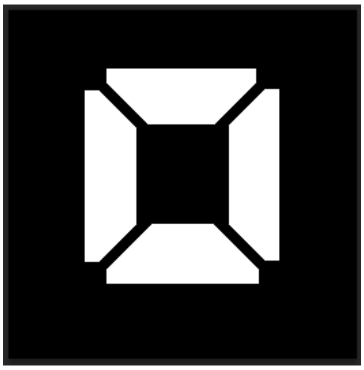
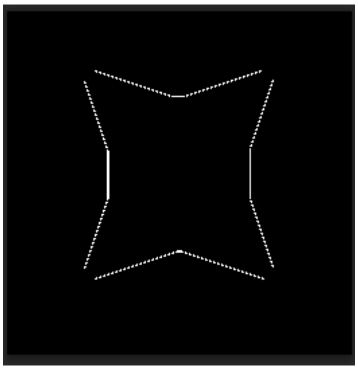
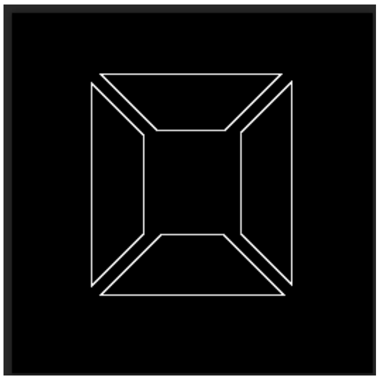
S(X) is the union of the **skeleton subsets** $\{S_n(X)\}$, $n = 0, 1, \ldots, N$,
where:

$$S_n(X) = (X \ominus nB) - (X \ominus nB) \circ B.$$

In my program, the value n is run from 0 to 20. That is, the erosion is performed on Sample2 for 20 times. I've also implemented the aforementioned basic functions like ImageUnion() and ImageDifference(), etc.

For (b), the boundary is extracted with a 3x3 full kernel. And the formula is really simple:
boundary = original - erosion(original)

For (c), the smoothed contour can be achieved through the "opening" morphology, where we use dilation after an erosion operation. A 11x11 kernel is used here.

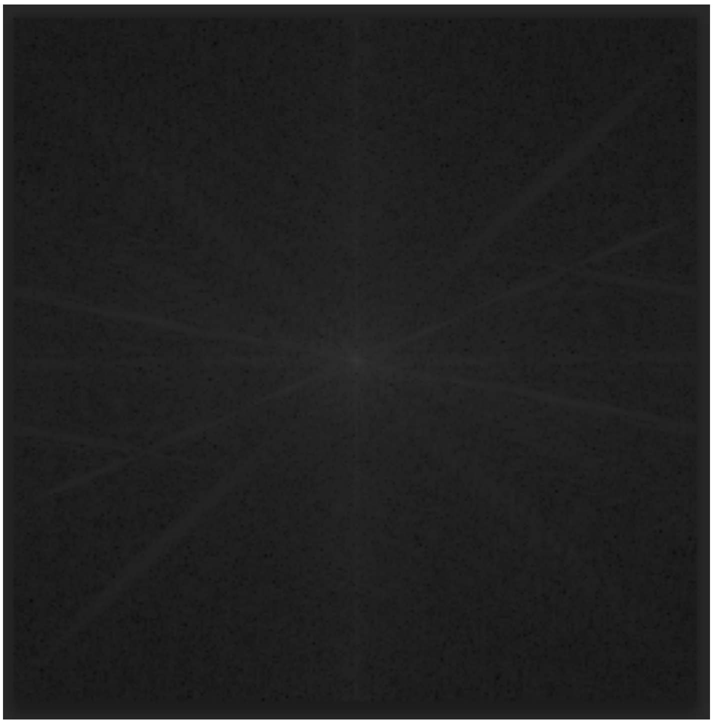




The following pictures are my results.

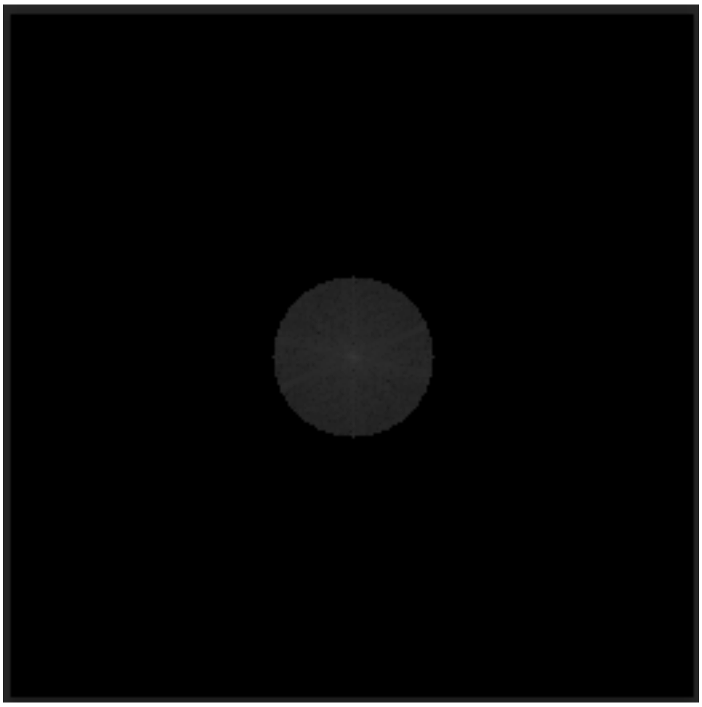
Problem 3

In the problem, the DFT formula is well described on the slides. Also, the filters are on the slides too. Here’s my result.

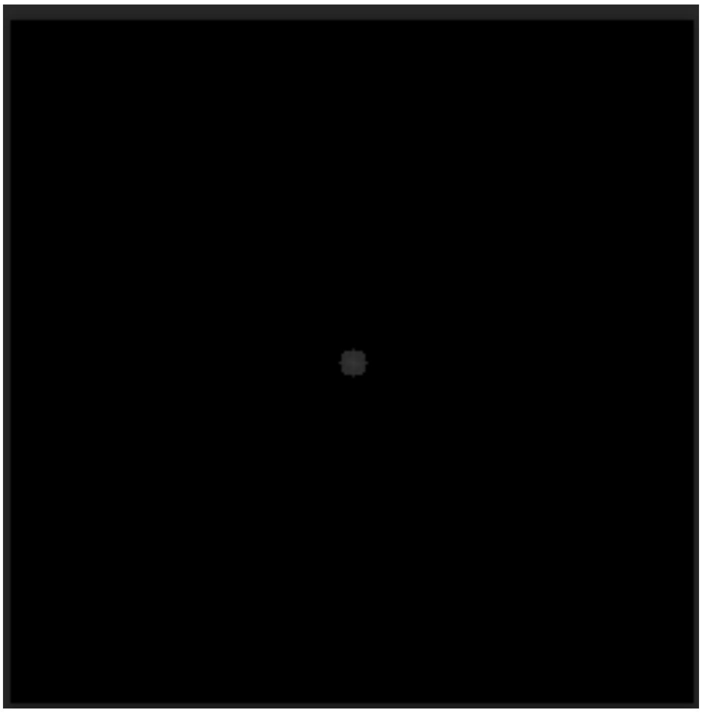
The DFT on Sample3:



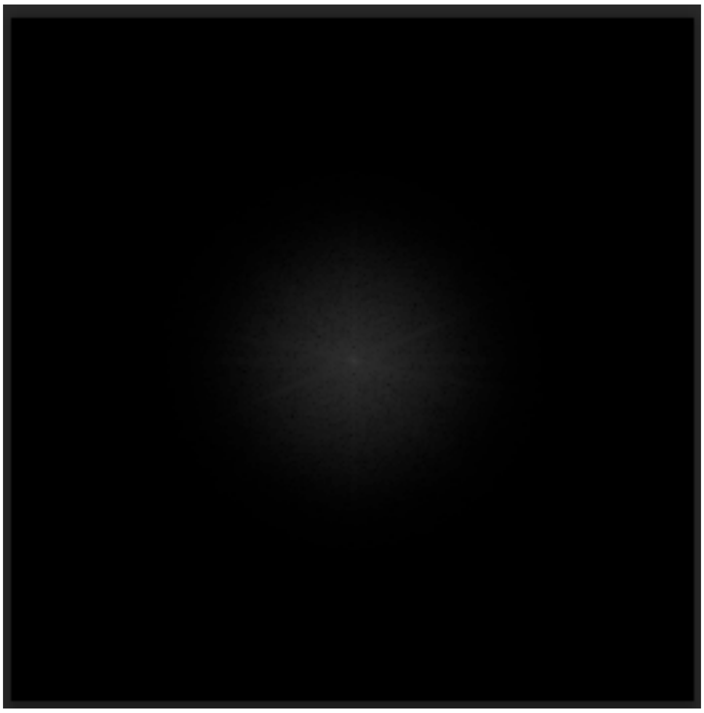
Ideal Low-Pass Filter with D0 = 30:



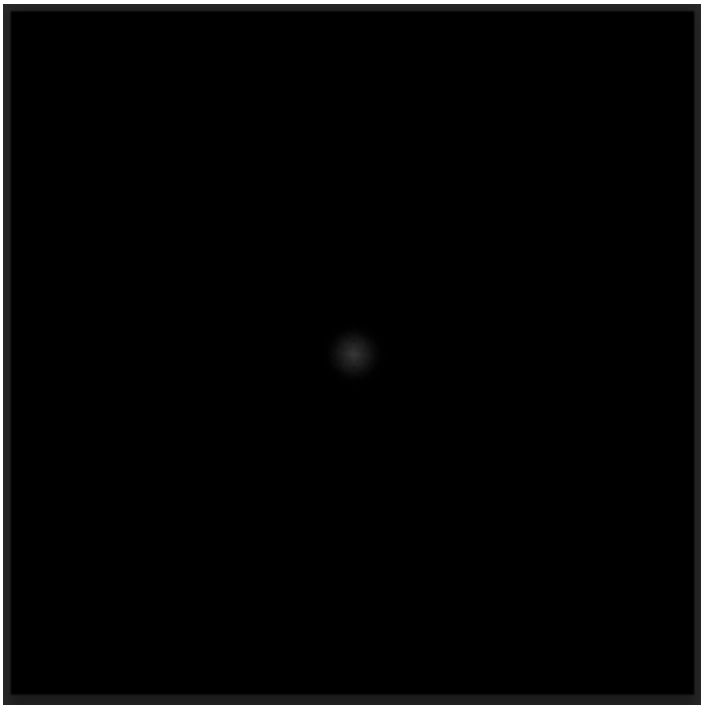
Ideal Low-Pass Filter with D0 = 5:



Gaussian Low-Pass Filter with D0 = 30:



Gaussian Low-Pass Filter with D0 = 5:



G_30 in spatial domain after applying inverse DFT: