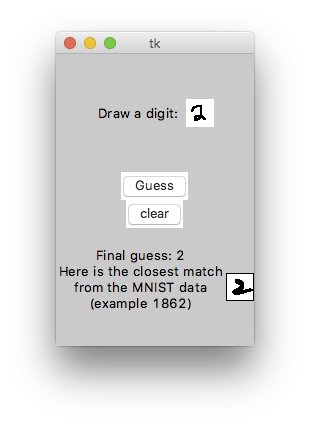
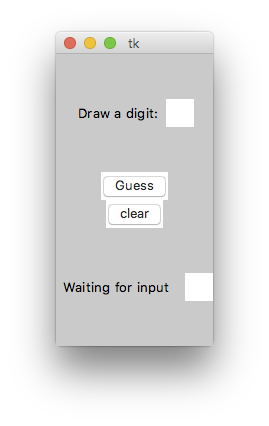
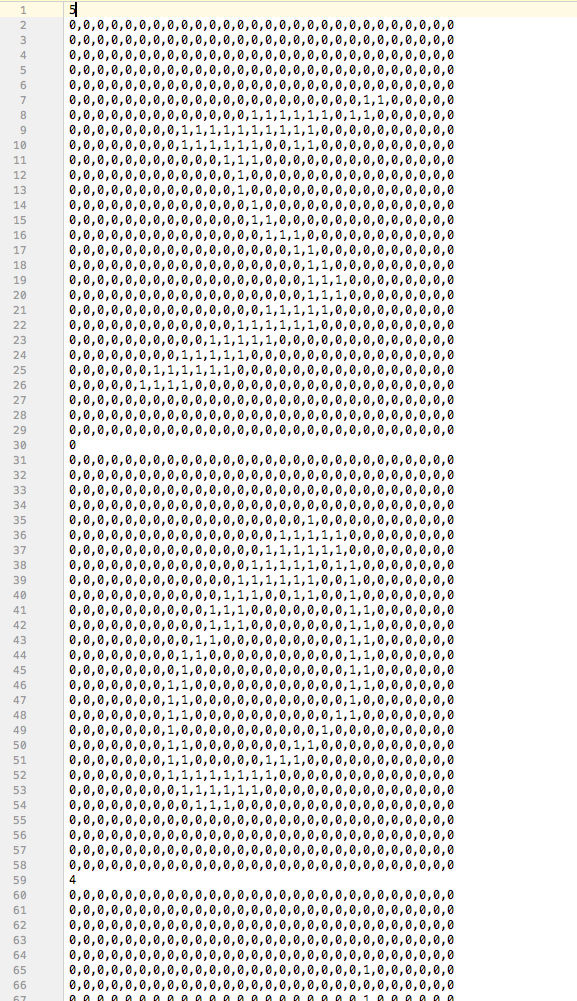
**Handwriting Recognition Project**

**Assignment**

You will write a GUI application that performs handwriting recognition for handwritten digits 0-9. Your application will allow the user to draw a character on a Canvas and then make a prediction for which digit you think they drew.

Handwriting recognition algorithms usually involve building models from many different handwritten samples collected from many different people. You will be implementing a simple, but powerful, algorithm called the Nearest Neighbor algorithm. The Nearest Neighbor algorithm works by comparing a newly-drawn character against all known samples and then making a guess based on the sample that is most similar to the newly-drawn character.

For our handwriting samples, we will use a data set which is based on a subset of the MNIST database (<http://yann.lecun.com/exdb/mnist/> ). The format of the mnist\_partial.csv file is shown below. Note that every block of 29 lines represents one handwritten digit. The first line in the block of 29 lines gives the digit being represented and the next 28 lines give the pixel values of the 28x28 image. A 0 pixel value represents a white color and a 1 represents black.



Points for this project will be awarded based on the following;

1. (3 points) Set up a GUI like the one above with *two* canvas widgets (one for drawing on and the other for displaying the image from the data set), a **Guess** button, and appropriate labels for prompting the user and displaying the guessed digit. You may include other components (like the clear button shown in the example), but they are not required.
2. (3 points) Your application allows the user to draw on one of the canvases using a click-drag event. An example of how to do this is given in the draw\_on\_canvas\_save\_as\_2D\_list.py sample code file.
3. (3 points) Your application captures the drawn image as a 2D list. An example of how to do this is given in the draw\_on\_canvas\_save\_as\_2D\_list.py sample code file.
4. (3 points) Your application can read data from the mnist\_partial.csv file, save it as a 2D list, and then display the appropriate character on the second canvas. An example of how to do this is given in the display\_characters\_from\_dataset.py sample code file.
5. (3 points) You correctly implement the Nearest Neighbor algorithm. When the Guess button is clicked, your application should do the following:

Loop through each handwriting sample in the mnist\_partial.csv file. For each one it should:

* 1. Read the sample into a 2D list
  2. Compare the sample 2D list against the 2D list you captured from the drawing and compute some kind of similarity score. I have had pretty good success by giving 5 points for each black pixel that matches between the two lists and 1 point for each white pixel that matches (and 0 points for each pixel that don’t match). You will probably need nested for loops, and I suggest putting it in its own function so you can pass the two lists as arguments and return the similarity score.
  3. Keep track of the score, the 2D list, and the digit it represents for the sample with the best similarity score you have seen so far.

Then after looking at all samples, display the digit (your guess) and the image for the sample with the best similarity score.

For an extra just-for-fun challenge, you can also try implementing a more complex version of the Nearest Neighbor prediction algorithm called *k-Nearest-Neighbor*. The value of *k* can be any integer greater than 1, and it works by keeping a list of the *k* best similarity scores and then predicting the digit which is most common among the *k* best.