In [11]:

```python
from sklearn.neighbors import KDTree

## Build nearest neighbor structure on training data
t_before = time.time()
kd_tree = KDTree(train_data)
t_after = time.time()

## Compute training time
t_training = t_after - t_before
print("Time to build data structure (seconds): ", t_training)

## Get nearest neighbor predictions on testing data
t_before = time.time()
test_neighbors = np.squeeze(kd_tree.query(test_data, k=1, return_distance=False))
kd_tree_predictions = train_labels[test_neighbors]
t_after = time.time()

## Compute testing time
t_testing = t_after - t_before
print("Time to classify test set (seconds): ", t_testing)

## Verify that the predictions are the same
print("KD tree produces same predictions as above? ", np.array_equal(test_predictions, kd_
tree_predictions))
```

```
Time to build data structure (seconds):  1.365771770477295
Time to classify test set (seconds):  30.783735990524292
KD tree produces same predictions as above?  True
```

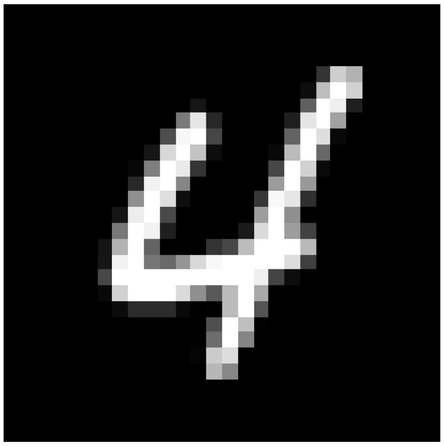
Week 1 - Programming Lab

Question 1 - Nearest Neighbor on MNIST

a)

In [12]:

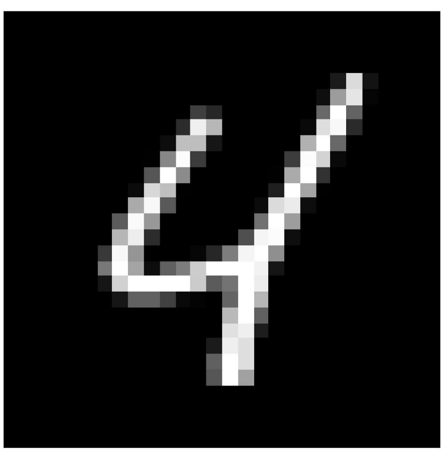
```
vis_image(100, "test") #image of test point 100
```



Label 4

In [13]:

```
vis_image(find_NN(test_data[100]), "train") #image of NN in training set
```



Label 4

The test point is classified correctly since it also labeled as four.

b)

In [14]:

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(test_predictions,test_labels)
print (cm)
```

```
[[ 99   0   0   0   0   1   0   0   2   1]
 [  0 100   1   0   0   0   0   4   0   1]
 [  0   0  94   2   0   0   0   0   1   1]
 [  0   0   1  91   0   0   0   0   1   1]
 [  0   0   0   2  97   0   0   1   1   2]
 [  1   0   0   4   0  98   1   0   0   1]
 [  0   0   0   0   0   0  99   0   1   0]
 [  0   0   3   0   0   0   0  94   1   3]
 [  0   0   1   1   0   0   0   0  92   0]
 [  0   0   0   0   3   1   0   1   1  90]]
```

The most misclassified digit is number 7 with three misclassification to 2, one to 1, and 3 to the digit 9. The least misclassified digit is 6 since we only have one misclassification to 8.

c)

In [133]:

```
# Create a dictionary to store the mean of each digit, empty vector
digit_means = {i: [] for i in range(10)}

# Iterate through each image
for i in range(len(train_data)):
    # Get the current image
    image = train_data[i, :]
    # Get the current label
    label = train_labels[i]
    # Append the image mean to the appropriate digit in the dictionary
    digit_means[label].append(np.mean(image))
```

In [134]:

```
# Print the mean of each digit
for digit in digit_means:
    print(f"Mean for digit {digit}: {np.mean(digit_means[digit])}")
```

```
Mean for digit 0: 43.821624755859375
Mean for digit 1: 19.2349910736084
Mean for digit 2: 38.428504943847656
Mean for digit 3: 35.87509536743164
Mean for digit 4: 31.09947395324707
Mean for digit 5: 33.41828918457031
Mean for digit 6: 34.772926330566406
Mean for digit 7: 29.32533836364746
Mean for digit 8: 38.497188568115234
Mean for digit 9: 31.085073471069336
```

Question 2 - Classifying back injuries

In [17]:

```
import numpy as np
# load data set and code labels as 0='no' , 1 = 'dh', 2= 'sl'
labels = [b'NO' , b'DH', b'SL']
data = np.loadtxt('spine-data.txt', converters= {6: lambda s: labels.index(s)})
```

In [100]:

```
training_Set = data[0:250]
training_labels = training_Set[:, -1]
```

In [101]:

```
test_Set = data[250:]
test_labels = test_Set[:, -1]
```

a)

In [135]:

```
from sklearn.neighbors import KNeighborsClassifier

# Create a KNeighborsClassifier object with L1 distance metric
knn = KNeighborsClassifier(n_neighbors=1, metric='l1')

# Fit the model on the training data
knn.fit(training_Set, training_labels)

# Make predictions on the test data
y_pred = knn.predict(test_Set)

# Calculate the number of incorrect predictions
incorrect = np.sum(y_pred != test_labels)

# Calculate the error rate
error_rate1 = incorrect / len(test_labels)
```

In [136]:

```
#error rate of l1
print(error_rate1)
```

```
0.18333333333333332
```

In [137]:

```
from sklearn.neighbors import KNeighborsClassifier

# Create a KNeighborsClassifier object with L2 distance metric
knn = KNeighborsClassifier(n_neighbors=2, metric='l2')

# Fit the model on the training data
knn.fit(training_Set, training_labels)

# Make predictions on the test data
y_pred = knn.predict(test_Set)

# Calculate the number of incorrect predictions
incorrect = np.sum(y_pred != test_labels)

# Calculate the error rate
error_rate2 = incorrect / len(test_labels)
```

In [138]:

```
#error rate of l2
print(error_rate2)
```

```
0.26666666666666666
```

The error rate for l1 is 0.1833, and for l2 is 0.266. The error rate is higher for l2 compared to l1.

b)

In [139]:

```
from sklearn.metrics import confusion_matrix
# Create the confusion matrix
cm = confusion_matrix(test_labels, y_pred)

# Print the confusion matrix
print(cm)
```

```
[[15  0  1]
 [13  5  0]
 [ 1  1 24]]
```

The most misclassified is the Hearniated disk with a normal spine. This means most people with herniated disk do not get diagnosed as having an abnormality.

In [ ]:

```

```