INTRODUCTION TO DEEP LEARNING
HOMEWORK 1

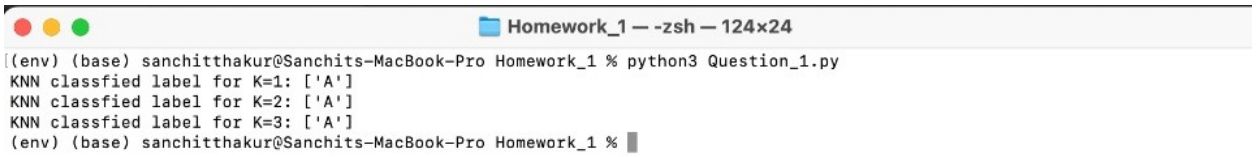
QUESTION 1:

The code written for the first question can be found below:

```
import numpy as np
import matplotlib as mpl
mpl.use('Agg')
import matplotlib.pyplot as plt

train_points = np.array([[0,1,0], [0,1,1], [1,2,1], [1,2,0], [1,2,2], [2,2,2],
[1,2,-1], [2,2,3], [-1,-1,-1], [0,-1,-2], [0,-1,1], [-1,-2,1]])
train_labels = np.array([0,0,0,0,1,1,1,1,2,2,2,2])
test = np.array([[1,0,1]])

labels_mapping = {0: 'A', 1: 'B', 2: 'C'}

# Define knn classifier
def kNNClassify(newInput, dataSet, labels, k):
   result=[]
   ########################
   # Input your code here #
   ########################
   for pt in newInput:
       distance=np.linalg.norm(dataSet-pt,axis=1)
       closest_indices=np.argsort(distance)[:k]          #Obtain the indices of the k
nearest neighbours
       closest_labels=[train_labels[i] for i in closest_indices]  #Obtain the labels
of the k nearest neighbours
       result_label = labels_mapping[np.argmax(np.bincount(closest_labels))]
       result.append(result_label)

   ####################
   # End of your code #
   ####################
   return result

outputlabel_k1 = kNNClassify(test,train_points,train_labels,1)
outputlabel_k2 = kNNClassify(test,train_points,train_labels,2)
```

```
outputlabel_k3 = kNNClassify(test,train_points,train_labels,3)

print('KNN classfied label for K=1:', outputlabel_k1)
print('KNN classfied label for K=2:', outputlabel_k2)
print('KNN classfied label for K=3:', outputlabel_k3)
```

The output obtained for this code is shown below:



As we can see from the above output, on repeated execution of the program, we receive the same output. Thus, the classified label for (1,0,1) for all scenarios of k=1,2 and 3 would be Class A.

QUESTION 2:

The code for this question can be found below:

```
import numpy as np
import matplotlib as mpl
mpl.use('Agg')
import matplotlib.pyplot as plt

# load mini training data and labels
mini_train = np.load('knn_minitrain.npy')
mini_train_label = np.load('knn_minitrain_label.npy')

# randomly generate test data
mini_test = np.random.randint(20, size=20)
mini_test = mini_test.reshape(10,2)



# Define knn classifier
def kNNClassify(newInput, dataSet, labels, k):
   result=[]
```

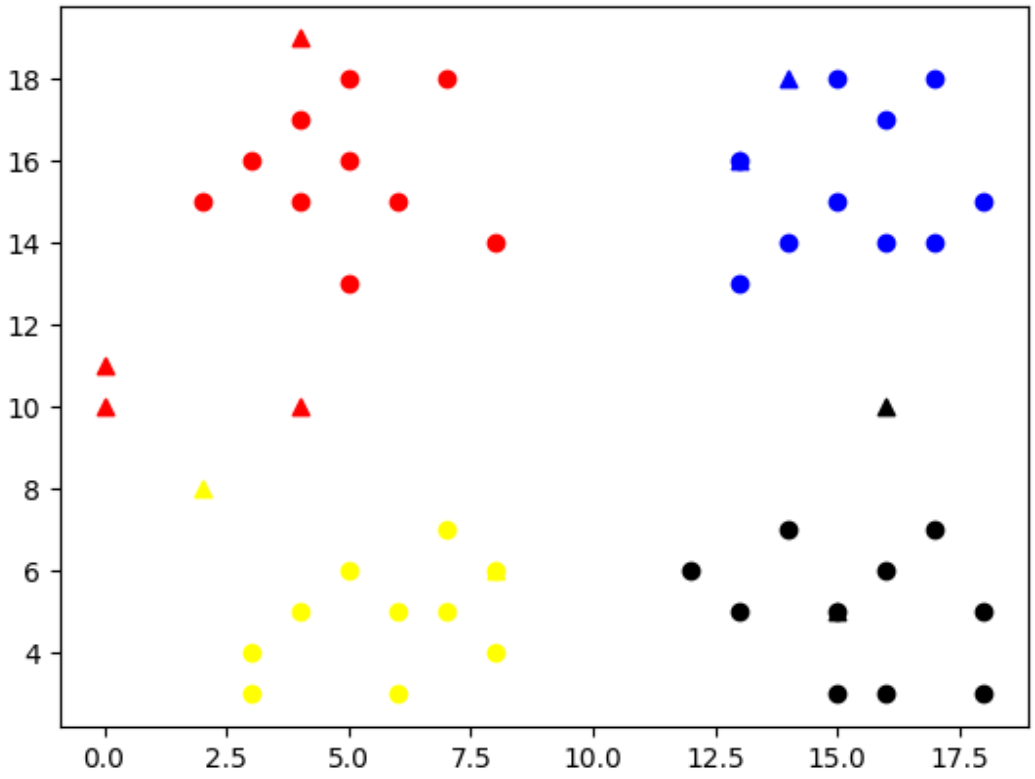
```
    ########################
    # Input your code here #
    ########################
    for pt in newInput:
        distance=np.linalg.norm(dataSet-pt,axis=1)
        closest_indices=np.argsort(distance)[:k]          #Obtain the indices of the k
nearest neighbours
        closest_labels=[labels[i] for i in closest_indices]  #Obtain the labels of the
k nearest neighbours
        result_label = np.argmax(np.bincount(closest_labels))
        result.append(result_label)

    ####################
    # End of your code #
    ####################
    return result

outputlabels=kNNClassify(mini_test,mini_train,mini_train_label,4)

print('random test points are:', mini_test)
print('knn classfied labels for test:', outputlabels)

# plot train data and classfied test data
train_x = mini_train[:,0]
train_y = mini_train[:,1]
fig = plt.figure()
plt.scatter(train_x[np.where(mini_train_label==0)],
train_y[np.where(mini_train_label==0)], color='red')
plt.scatter(train_x[np.where(mini_train_label==1)],
train_y[np.where(mini_train_label==1)], color='blue')
plt.scatter(train_x[np.where(mini_train_label==2)],
train_y[np.where(mini_train_label==2)], color='yellow')
plt.scatter(train_x[np.where(mini_train_label==3)],
train_y[np.where(mini_train_label==3)], color='black')

test_x = mini_test[:,0]
test_y = mini_test[:,1]
outputlabels = np.array(outputlabels)
plt.scatter(test_x[np.where(outputlabels==0)], test_y[np.where(outputlabels==0)],
marker='^', color='red')
plt.scatter(test_x[np.where(outputlabels==1)], test_y[np.where(outputlabels==1)],
marker='^', color='blue')
```

```
plt.scatter(test_x[np.where(outputlabels==2)], test_y[np.where(outputlabels==2)],
marker='^', color='yellow')
plt.scatter(test_x[np.where(outputlabels==3)], test_y[np.where(outputlabels==3)],
marker='^', color='black')

#save diagram as png file
plt.savefig("miniknn.png")
```

The output for this question can be found below:

```
Homework_1 — -zsh — 124×24
(env) (base) sanchitthakur@Sanchits-MacBook-Pro Homework_1 % python3 Question_2.py
random test points are: [[14 18]
 [ 4 10]
 [16 10]
 [15  5]
 [ 8  6]
 [ 0 11]
 [ 2  8]
 [ 4 19]
 [ 0 10]
 [13 16]]
knn classfied labels for test: [1, 0, 3, 3, 2, 0, 2, 0, 0, 1]
(env) (base) sanchitthakur@Sanchits-MacBook-Pro Homework_1 %
```

The graph visualization of the result is shown below:



As we can see, the 10 randomly generated 2-dimension test data have been classified into appropriate labels. Also, as we can see, the value of k that has been chosen is 4.

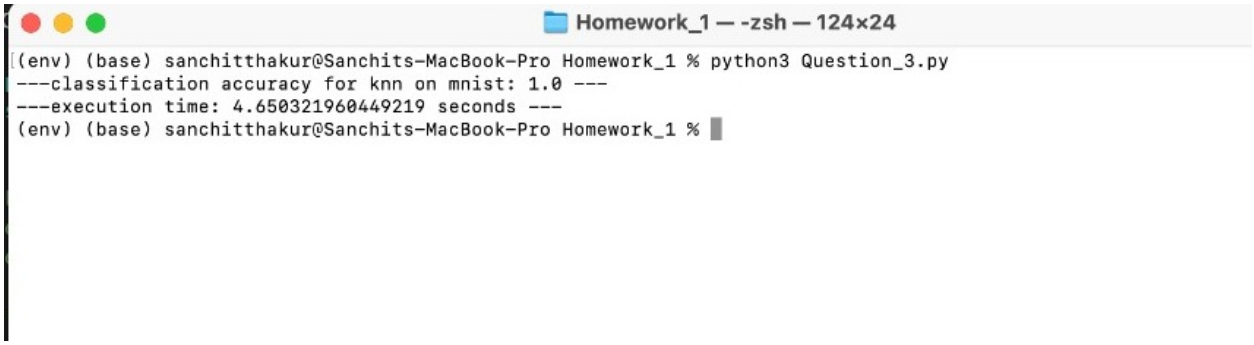
QUESTION 3:

The code for this question can be found below:

```
import math
import numpy as np
from download_mnist import load
import operator
import time
# classify using kNN
#x_train = np.load('../x_train.npy')
#y_train = np.load('../y_train.npy')
#x_test = np.load('../x_test.npy')
#y_test = np.load('../y_test.npy')
x_train, y_train, x_test, y_test = load()
x_train = x_train.reshape(60000,28,28)
x_test  = x_test.reshape(10000,28,28)
```

```
x_train = x_train.astype(float)
x_test = x_test.astype(float)
def kNNClassify(newInput, dataSet, labels, k):
   result=[]
   ########################
   # Input your code here #
   ########################

   for img in newInput:
       distance=np.linalg.norm(np.subtract(dataSet,img),axis=(1,2))
       closest_indices=np.argsort(distance)[:k]            #Obtain the indices of the
k nearest neighbours
       closest_labels=[labels[i] for i in closest_indices]  #Obtain the labels of the
k nearest neighbours
       result_label=np.argmax(np.bincount(closest_labels))
       result.append(result_label)

   ####################
   # End of your code #
   ####################
   return result

start_time = time.time()
outputlabels=kNNClassify(x_test[0:20],x_train,y_train,10)
result = y_test[0:20] - outputlabels
result = (1 - np.count_nonzero(result)/len(outputlabels))
print ("---classification accuracy for knn on mnist: %s ---" %result)
print ("---execution time: %s seconds ---" % (time.time() - start_time))
```

The output for the above code can be found below:



As we can see, we have achieved a classification accuracy of 1.0 and the time taken on this system is 4.65 seconds. Also, the value of k for this question has been chosen to be 10.