For this homework, you will be implementing the outlier detection algorithm from class.

There are two subtasks:

Task 1:

For this task, you will be asked to implement (in Python) the nested loops algorithm on slide 12 of the "outliers" lecture. Start with the code below, and fill in the appropriate missing code. To implement the priority queue, use Python's heapq. The distance between data points will be implemented using Euclidean distance (the l2-norm).

My output is:

```
>>> print("--- %s seconds ---" % (time.time() - start_time))
--- 79.09023690223694 seconds ---
>>>
>>> for outlier in outliers:
...     print (outlier)
...
(21.240308803962975, 3002)
(22.400311356557985, 3001)
(25.074870468424841, 3005)
(24.30004896822501, 3003)
(23.902326257409541, 3004)
>>>
```

Note that since the data are randomly created, you may get a slightly different result, although data points 3001 thru 3004 should always be the outliers. Since 79 seconds is a non-trivial amount of time to wait, when you are debugging, you should consider reducing the data set size from 3000 to a smaller value, such as 100. Here is the skeleton to start with:

```
import heapq as hq
import numpy sa np
import time

#create the covariance matrix
covar = np.zeros ((100,100))
np.fill_diagonal (covar, 1)

#and the mean vector
mean = np.zeros (100)

#create 3000 data points
```

```
all_data = np.random.multivariate_normal (mean, covar, 3000)

#now create the 20 outliers
for i in range (1, 20):
    mean.fill (i)
    outlier_data = np.random.multivariate_normal (mean, covar, i)
    all_data = np.concatenate ((all_data, outlier_data))

#k for kNN detection
k = 10

#the number of outliers to return
m = 5

#start the timer
start_time = time.time()

#the priority queue of outliers
outliers = []

#YOUR CODE HERE!

print("--- %s seconds ---" % (time.time() - start_time))

#print the outliers...
for outlier in outliers:
    print (outlier)
```

Task 2

In this task, you should implement the faster algorithm on slide 14. Note that to randomly shuffle the data you should use

```
np.random.shuffle (all_data)
```

Here is my output:

```
>>> print("--- %s seconds ---" % (time.time() - start_time))
--- 5.156224012374878 seconds ---
>>>
... for outlier in outliers:
...     print (outlier)
...
```

(21.240308803962975, 2085)
(22.400311356557985, 198)
(24.30004896822501, 3162)
(25.074870468424841, 51)
(23.902326257409541, 50)

Note that since you've shuffled the data, the indices of the outliers will change, but the distances should be the same.

Copy and paste your code and results to submit. You can either attach a file (text or PDF) or else directly copy and paste into Canvas.

IMPORTANT: DO NOT RE-CREATE THE DATA BETWEEN TASK 1 AND TASK 2. When we grade this (and when you debug), we want the set of outliers to stay the same. If you recreate the data, it is difficult to grade because the outliers change.