Q1) 2SUM

Problem : For an array A[1..n], output two different indices $1 \le p < q \le n$ such that $A[p] = -A[q]$ if exist, and "-1" otherwise. **Solution :** Sort the array first. Take two pointer one from the start and another from the end.

```
Loop :
Calculate the sum of the numbers stored in the pointers.
If sum is 0, then STOP.
Otherwise, if abs(*pointer1) < abs(*pointer2), then, pointer1 = pointer1 + 1
        otherwise, pointer2 = pointer2 - 1
```

Q2) Counting Inversions

Problem : Given a positive integer $n \le 10^5$ and an array A[1..n] of integers from -10^5 to 10^5, count number of inversions, which shows how far the array is from being sorted.
Solution : We will use the **Merge Sort** approach to solve this problem. In merge sort, we divide the array in two halves, and solve the problem for each half and merge them. Suppose, we know the count of inversions in each half of the array, then -

```
Total no. of inversions = no. of inv. in left half
                        + no. of inv. in right half
                        + no. of inv. to merge them
```

To count the number of inversions in each half -

```
In merge process, let i is used for indexing left sub-array and j for right sub-array.
At any step in merge(), if a[i] is greater than a[j], then there are (mid – i) inversions.
Because left and right subarrays are sorted, so all the remaining elements in left-subarray
(a[i+1], a[i+2] ... a[mid]) will be greater than a[j].
```

Q3) K-Means

Problem : Implement K-Means clustering for points in 5-D spaces, where origins are given initially.
Solution :

```
Loop until converges:
for each point in the 5-D space,
  calculate eucledian distance b/w the point and all the origins
  select the minimum distance from the origins and assign the corresponding cluster to it
Compute the new origins (mean)
```

Q4) Heirarchical Clustering

Problem : Given points in 5-D and type of linkage and clustering threshold value, implement Heirarchical

Clustering.

Solution :

```
Genrate the distance (eucledian) matrix.
Loop upto threshold value:
  Find the pair with minimum distance.
  Join them and form a cluster.
  if distance > threshold:
    STOP
  Compute the distance of the new cluster from rest of the points/clusters.
  if COMPLETE_LINKAGE :
    Take the maximum of all the distances
  else (SINGLE_LINKAGE) :
    Take the minimum of all the distances
```

Q5) Implement HMM.

Problem : Given the probability of symbol output and probability of state transition, create the table for all possibilities for the given input.

Solution : Generate all the state sequence recursively and and compute the probabilities of each individual state seq. and normalize the probability values (Prob-1). Then, to find the optimal sequence in the HMM sense, we choose the most probable state symbol at each position of the observeation sequence. To do this, we start with summing up the probabilities in Prob-1 that have an S1 (State-1) in the first position. Doing so, we find the (normalized) probability of S1 in the first position. We do it for each state and choose the state with highest probability value as most probable state at the first position. We repeat the process for each position of the observation seq. and get the second table consisting HMM Probabilities.