Question 3 :

The goal of the algorithm is to find the longest monotonically increasing subsequence of a sequence of n numbers. To do this, first we'll start we'll use two arrays M and P such that;

M[j] will store the index k of the smallest value X[k] such that there is an increasing subsequence of length j ending at X[k] on the range $k \leq i$.

And

P[k] will store the index of the predecessor of X[k] in the longest increasing subsequence ending at X[k]

Pseudo-code :

```
def longestIncreasingSubsequence(sequence, subsequence, elem):
        P = array of length N
        M = array of length N + 1

        L = 0
        for i in range 0 to N-1:
          // Binary search
              lo = 1
              hi = L
          while lo ≤ hi:
              mid = ceil((lo+hi)/2)
               if sequence[M[mid]] < X[i]:
                     lo = mid+1
            else:
                     hi = mid-1

          // After searching, lo is 1 greater than the
          // length of the longest prefix of sequence[i]
          newL = lo

          // The predecessor of sequence[i] is the last index of the subsequence of
       length newL-1
          P[i] = M[newL-1]
          M[newL] = i

         if newL > L:
          L = newL
```

```
Solution = array of length L
k = M[L]
for i in range L-1 to 0:
        Solution[i] = X[k]
        k = P[k]

return Solution
```

The binary search is be done in time O(log n) since the subsequences are in increasing order . The loop will run O(n) times, resulting in a running time of O(n log n).

Help from wikipedia page.

Question 4 :

W1 = AACT && W2 = GAT. The optimal alignment with a minimum score of 0 using the edit score is :

AACT

GA-T

d(x,y) = {-1 if x = y

{ 1 otherwise.

	-	A	A	C	T
-	0	1	2	3	4
G	1	1	2	3	4
A	2	0	0	1	2
T	3	1	1	1	0