Quiz3B (COL 351)

Name **Entry No.**

Give precise arguments. If using dynamic programming, you must clearly state what each entry of the table denotes, and in which order to compute the entries. Also give an explanation of the recurrence used.

A subsequence of a string $x = x_1, x_2, \ldots, x_n$ is a subset of characters $x_{i_1}, x_{i_2}, \ldots, x_{i_k}$ such that $i_1 < i_2 < \ldots < i_k$. Such a subsequence is said to be palindromic if it is the same whether read left to right or right to left. For instance, the string $A, C, G, T, G, T, C, A, A, A, C, A, T, C, G$ has many palindromic subsequences, including A, C, G, C, A and A, A, A, A (on the other hand, the subsequence A, C, T is not palindromic). Devise an algorithm that takes a string $x[1 \ldots n]$ and returns the **length** of the longest palindromic subsequence in it. Its running time should be $O(n^2)$.

Solution We build a 2-dimensional table T, where the entry $T[i, j]$ stores the longest palindrome in the string $x_i, \ldots, x_j$. Now we write a recursive definition of $T[i, j]$. If symbols x_i and x_j are same, we can assume that they are part of the longest palindrome — this is so because of the following reason. If neither x_i nor x_j are part of the longest palindrome of $x_i, \ldots, x_j$, then we can add both of these characters to get a longer palindrome. If only one of them, say x_i is part of it, and is matched with x_k, then we can get another palindrome of equal length by replacing x_k by x_j. So, $T[i, j]$ will be equal to $T[i + 1, j - 1]$. If the two symbols are different, then they both cannot be part of any palindrome . So, $T[i, j]$ will be maximum of $T[i + 1, j]$ and $T[i, j - 1]$. Thus, we get

$$T[i,j] = \begin{cases} T[i+1, j-1] & \text{if } x_i = x_j \\ \max(T[i+1, j], T[i, j-1]) & \text{otherwise} \end{cases}$$

The base case happens when $i = j$, in which case the answer is 1 . The for loops are :

```
for i = 1 to n
    T[i,i] = 1;
for i=n-1 downto 1
   for j = i+1 to n
      T[i,j] = as mentioned above
```

The algorithm returns $T[1, n]$. Clearly the running time is $O(n^2)$.