LAB SESSION 07:

Date of the Session:___/___/___ **Time of the Session:____to____**

Pre-Lab:

1) A student named Satish is eagerly waiting to attend tomorrow's class. As he searched the concepts of tomorrow's lecture in the course handout and started solving the problems, in the middle he was stroked in the concept of strings because he was poor in this concept so help him so solve the problem, given two strings str1 and str2 and below operations that can performed on str1. Find minimum number of edits (operations) required to convert 'str1' into 'str2'.

1. Insert
2.Remove
3.Replace

Input
str1 = "cat", str2 = "cut"
Output
1
We can convert str1 into str2 by replacing 'a' with 'u'.
Input
str1 = "sunday", str2 = "saturday"
Output
3

```
def editDistance(str1, str2, m, n)
if m == 0:
    return n
if n == 0:
    return m
if str1[m-1] == str2[n-1]
    return editDistance(str1, str2, m-1, n-1)
return 1 + min(editDistance(str1, str2, m, n-1),
               editDistance(str1, str2, m-1, n),
               editDistance(str1, str2, m-1, n-1))
str1 = "sunday"
str2 = "saturday"
Print(editDistance(str1, str2, len(str1), len(str2)))
```

19003075g

2) Given N numbers n1,n2,...nN and Q queries q1,q2,...qQ. Your task is to print Q (Q< j <numbers) f1,f2,...fQ, corresponding to query qj1 max(n1=fj ,n2 ,...nq) using dynamic programming.

Input

5 3
5 4 8 6 9
2 3 5

Output

5 8 9

```
Class Solution:
    def findQuery(self, nums, ques):
        for i in range(int(input())):
            n,q = map(int, input().split())
            lst1 = list(map(int, input().split()))
            lst2 = list(map(int, input().split()))
            j = 0
            for i in range(q):
                x = lst2[j]
                y = lst2[j+1]
                k = lst2[j+2]
                j+ = 3
                l = []
                l = lst[x-1:y]
                l.sort()
                print(l)[k-1], end = [" "]
                print()
```

INPUT: 5 3
5 4 8 6 9
2 3 5

OUTPUT: 5 8 9

190030259

In-Lab:

1) Bhanu is a student at KL University who likes playing with strings, after reading a question from their lab workbook for the ADA Course she found what is meant by a subsequence.

(A subsequence is a sequence that can be derived from another sequence by deleting some or no elements without changing the order of the remaining elements)

So, she created 2 strings out of which one she considered as a master string and the other one as a slave string. She challenged her friend Teju to find out whether the slave string is a subsequence of the master string or not, As Teju is undergoing her CRT classes she decided to code the logic for this question. Help her in building the logic and write a code using Dynamic programming concept.

```
def CRT[x,y,m,n]:
    if m==0 & n==0
     return 0
    elif x[m-1]==y[n-1]:
    return 1+CRT[x,y,m-1,n-1]
    else:
    return max(CRT(x,y,m-1,n-1)
                CRT(x,y,m-1,n)):
     X="AGGTAB"
     Y="GXTYAYB"
    Print("length of CRT is", CRT(X,Y,len(x),len(y))
```

OUTPUT:

length of CRT is 4.

190030759

2) Geeta is working in a company, and she has n different projects to work on, where every project is scheduled to be done from startTime[i] to endTime[i], obtaining a profit of profit[i]. You are given the startTime, endTime and profit arrays, return the maximum profit you can take such that there are no two projects that she is working on in that given subset with overlapping time range. If she chooses a project that ends at time a then she will be able to start another project that starts at time b.

Input
startTime = [1,2,3,4,6], endTime = [3,5,10,6,9], profit = [20,20,100,70,60]
Output
150
Explanation: The subset chosen is the first, fourth and fifth project.
Profit obtained 150 = 20 + 70 + 60.

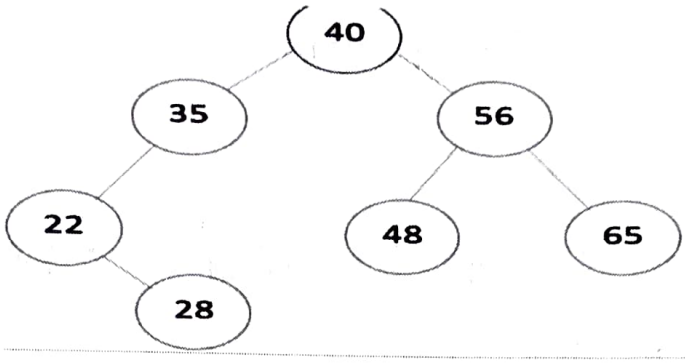
```
def jobScheduling(self, startTime: List[int], endTime: List[int],
                  profit: List[int]) -> int:
    import bisect
    info = [(start, end, p) for start, end, p in zip(startTime, profit,
                                                     endTime)]
    endTime = sorted(endTime)
    n = len(info)
    dp = [0 for in range(n)]
    dp[0] = info[0][2]
    for i in range(1,n):
    pos = bisect.bisect(endTime[:i], info[i][0])
     if pos == 0:
     dp[i] = max(dp[i-1], info[i][2])
     else:
      dp[i] = max(dp[i-1], info[i][2]+dp[pos-1])
       return dp[i]
```

INPUT: [1,2,3,4,6]
[3,5,10,6,9]
[20,20,100,70,60]

OUTPUT: 150

190030759

3) Teacher: good morning students!!!
Students: good morning mam
Teacher: Today topic is binary tree. Does anyone know what is binary tree?
Students: no mam
Teacher: In computer science, a binary tree is a tree data structure in which each node has at most two children, which are referred to as the left child and the right child. Satish you completed this topic yesterday can you explain this topic?
Satish: Sure, mam but I got doubt this concept, do anyone help me in solving this the question was Given a binary tree, find whether if a given Binary Tree is Balanced?




```
Class Node:
  def __init__(self,key,left=None,right=None):
        self.key = key
        self.left = left
        self.right=right
def is Height balanced (root, isBalanced =True):
   if root is None not is Balanced:
        return 0, is Balanced
left_height, is Balanced = is Height Balanced (root.left is Balanced)
right_height, is Balanced = is Height Balanced (root.Right, is Balanced)
if abs(left.height_right height) >1:
        is Balanced =False
return max(left_height, right_height)+1 is Balanced
if __name__ == "__main__":
   root= Node(40)
   root.left = Node(35)
```

```
root.right=Node(56)
root.left_left=Node(22)
root.right.left=Node(48)
root.right.right=Node(65)
root.left.left.right=Node(28)
if is Height Balanced (root)[1]:
    Print("Binary Tree is Balanced")
else:
    Print("Binary Tree is Not Balanced")
```

Output: Binary Tree is not Balanced.

190030759

Post-Lab:

1) SIRI studies at KL University and a person who is interested in Dynamic Programming, she created a question for you to solve. she decided to give a question related to palindrome, you need to use dynamic programming to solve this problem brute force is not allowed since she hates waiting too long to find the answer. The question follows like this find the longest palindromic subsequence.

(Unlike substrings, subsequences are not required to occupy consecutive positions within the original string.)

Input

ABBDCACB

Output

The length of the longest palindromic subsequence is 5

The longest palindromic subsequence is BCACB

```
def max(x,y):
    if (x>y):
        return x
    return y
def lps(seq, i, j):
    if (i==j)
        return 1
    if (seq[i]==seq[j] and i+1==j):
        return 2
    if (seq[i]==seq[j]):
        return lps(seq, i+1, j-1)+2
    return max(lps(seq, i, j-1),
               lps(seq, i+1, j))
if __name__=="__main__":
    seq="ABBDCACB"
    n=len(seq)
print("The length of the lps is," lps(seq,0,n-1))
```

```
def longest palsub seq (string):
    rev = string [::-1]
    return max (string, rev)
if __name__ = "__main__":
    String = "ABBDCACB":
Print (long espal subseq (string))
```

INPUT: ABBDCACB

OUTPUT: The length of the longest palindromic subsequence is 5

~~The length of the~~

The longest palindromic subsequence is BCACB

1900307S9

2) Bhanu and Teju are playing dice game where there are N dice with M faces and the dice are numbered from 1 to M. A person wins the game if the sum of the faces of dice adds up to a value X. you are playing on Bhanu's team, and It is Teju's turn now.
You are supposed to find number of ways your opponent can win the game where N, M and X are provided as input. Use Dynamic programming to solve the problem.
Using DP (Dynamic programming) to find the number of ways to get sum X.

Input
M = 2
N = 2
X = 3
Output
2

```
def findways(m,n,x):
    table=[[0]*(x+1) for i in range(n+1)]
for j in range(1,min(m+1,x+1)):
    table[i][j]=1
for i in range(2,x+1):
for j in range(1,x+1):
    for k in range(1,min(m+1,j)):
    table[i][j]+=table[i-1][j-k]
    return table[-1][-1]
    print(findways(2,2,3))
```

INPUT: M=2
N=2
x=3

OUTPUT: 2

(For Evaluator's use only)

Comment of the Evaluator (if Any)	Evaluator's Observation
	MarksSecured: outof
	Full Name of the Evaluator:
	Signature of the Evaluator Date of Evaluation: