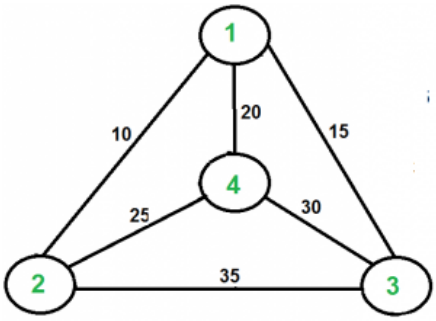
Started on	Thursday, 8 May 2025, 11:18 AM
State	Finished
Completed on	Thursday, 8 May 2025, 11:54 AM
Time taken	35 mins 18 secs
Grade	**80.00** out of 100.00

Question **1**

Correct

Mark 20.00 out of 20.00

Solve Travelling Sales man Problem for the following graph



Answer: (penalty regime: 0 %)

Reset answer

```
from sys import maxsize
from itertools import permutations
V = 4
def travellingSalesmanProblem(graph, s):
    vetex=[]
    cur=0
    minpath=maxsize
    for i in range(V):
        if i!=s:
            vetex.append(i)
    nextper=permutations(vetex)
    for i in nextper:
        cur=0
        k=s
        for j in i:
            cur+=graph[k][j]
            k=j
        cur+=graph[k][s]
        minpath=min(minpath,cur)
    return minpath
if __name__ == "__main__":
    graph = [[0, 10, 15, 20], [10, 0, 35, 25],[15, 35, 0, 30], [20, 25, 30, 0]]
```

	Expected	Got	
✔	80	80	✔

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.

Question **2**

Not answered

Mark 0.00 out of 20.00

LONGEST PALINDROMIC SUBSEQUENCE

Given a sequence, find the length of the longest palindromic subsequence in it.

For example:

Input	Result
ABBDCACB	The length of the LPS is 5

Answer: (penalty regime: 0 %)

```
1
```

Question **3**

Correct

Mark 20.00 out of 20.00

Create a python program for 0/1 knapsack problem using naive recursion method

For example:

Test	Input	Result
knapSack(W, wt, val, n)	3 3 50 60 100 120 10 20 30	The maximum value that can be put in a knapsack of capacity W is: 220

Answer: (penalty regime: 0 %)

Reset answer

```
def knapSack(W, wt, val, n):
    if W==0 or n==0:
        return 0
    if wt[n-1]>W:
        return knapSack(W,wt,val,n-1)
    else:
        inc=val[n-1]+knapSack(W-wt[n-1],wt,val,n-1)
        exc=knapSack(W,wt,val,n-1)
        return max(inc,exc)

x=int(input())
y=int(input())
W=int(input())
val=[]
wt=[]
for i in range(x):
    val.append(int(input()))
for y in range(y):
    wt.append(int(input()))
n = len(val)
print('The maximum value that can be put in a knapsack of capacity W is: ',knapSack(W, wt, val, n))
```

	Test	Input	Expected	Got	
✔	knapSack(W, wt, val, n)	3 3 50 60 100 120 10 20 30	The maximum value that can be put in a knapsack of capacity W is: 220	The maximum value that can be put in a knapsack of capacity W is: 220	✔
✔	knapSack(W, wt, val, n)	3 3 55 65 115 125 15 25 35	The maximum value that can be put in a knapsack of capacity W is: 190	The maximum value that can be put in a knapsack of capacity W is: 190	✔

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.

Question **4**

Correct

Mark 20.00 out of 20.00

Create a python program to for the following problem statement.

You are given an `n x n grid` representing a field of cherries, each cell is one of three possible integers.

- `0` means the cell is empty, so you can pass through,
- `1` means the cell contains a cherry that you can pick up and pass through, or
- `-1` means the cell contains a thorn that blocks your way.

Return *the maximum number of cherries you can collect by following the rules below*:

- Starting at the position `(0, 0)` and reaching `(n - 1, n - 1)` by moving right or down through valid path cells (cells with value `0` or `1`).
- After reaching `(n - 1, n - 1)`, returning to `(0, 0)` by moving left or up through valid path cells.
- When passing through a path cell containing a cherry, you pick it up, and the cell becomes an empty cell `0`.
- If there is no valid path between `(0, 0)` and `(n - 1, n - 1)`, then no cherries can be collected.

For example:

Test	Result
`obj.cherryPickup(grid)`	5

Answer: (penalty regime: 0 %)

Reset answer

```
class Solution:
    def cherryPickup(self, grid):
        n = len(grid)
        rows=len(grid)
        cols=len(grid[0])
        memo={}
        def dp(r,c1,c2):
            if r==rows or c1<0 or c1==cols or c2<0 or c2==cols:
                return 0
            if (r,c1,c2) in memo:
                return memo[(r,c1,c2)]
            cherries=grid[r][c1]+(grid[r][c2] if c1!=c2 else 0)
            maxcherries=0
            for dc1 in [-1,0,1]:
                for dc2 in [-1,0,1]:
                    maxcherries=max(maxcherries,dp(r+1,c1+dc1,c2+dc2))
            result=cherries+maxcherries
            memo[(r,c1,c2)]=result
            return result
        return dp(0,0,cols-1)
obj=Solution()
grid=[[0,1,-1],[1,0,-1],[1,1,1]]
```

	Test	Expected	Got	
✔	`obj.cherryPickup(grid)`	5	5	✔

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.

Question **5**

Correct

Mark 20.00 out of 20.00

Create a python program using brute force method of searching for the given substring in the main string.

For example:

Test	Input	Result
match(str1,str2)	AABAACAADAABAABA AABA	Found at index 0 Found at index 9 Found at index 12

Answer: (penalty regime: 0 %)

Reset answer

```
def match(string,sub):
    l = len(string)
    ls = len(sub)
    start = sub[0]
    for i in range(l-ls+1):
        j=0
        while j<ls and string[i+j]==sub[j]:
            j+=1
        if j==ls:
            print('Found at index',i)
    return -1
str1=input()
str2=input()
```

	Test	Input	Expected	Got	
✔	match(str1,str2)	AABAACAADAABAABA AABA	Found at index 0 Found at index 9 Found at index 12	Found at index 0 Found at index 9 Found at index 12	✔
✔	match(str1,str2)	saveetha savee	Found at index 0	Found at index 0	✔

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.