Started on	Thursday, 22 May 2025, 2:30 PM
State	Finished
Completed on	Thursday, 22 May 2025, 2:48 PM
Time taken	17 mins 46 secs
Grade	**100.00** out of 100.00

Question **1**

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)
        dp = [[0]*n for _ in range(n)]
        for i in range(n-1,-1,-1):
            for j in range(n-1, -1, -1):
                if i==n-1 and j==n-1:
                    dp[i][j] = grid[i][j]
                elif i==n-1:
                    dp[i][j] = grid[i][j]+dp[i][j+1]
                elif j==n-1:
                    dp[i][j] = grid[i][j]+dp[i+1][j]
                else:
                    dp[i][j] = grid[i][j]+max(dp[i][j+1], dp[i+1][j])
        return dp[0][0] + 1


obj=Solution()
grid=[[0,1,-1],[1,0,-1],[1,1,1]]
print(obj.cherryPickup(grid))
```

	Test	Expected	Got	
✓	obj.cherryPickup(grid)	5	5	✓

Passed all tests! ✓

Correct

Marks for this submission: 20.00/20.00.

Question **2**

Correct

Mark 20.00 out of 20.00

Create a python program to find the maximum value in linear search.

For example:

Test	Input	Result
`find_maximum(test_scores)`	10 88 93 75 100 80 67 71 92 90 83	`Maximum value is  100`

Answer: (penalty regime: 0 %)

Reset answer

```python
def find_maximum(lst):
    max=None
    for i in lst:
        if max== None or i>max:
            max=i
    return max

test_scores = []
n=int(input())
for i in range(n):
    test_scores.append(int(input()))

print("Maximum value is ",find_maximum(test_scores))
```

	Test	Input	Expected	Got	
✓	`find_maximum(test_scores)`	10 88 93 75 100 80 67 71 92 90 83	`Maximum value is  100`	`Maximum value is  100`	✓

	Test	Input	Expected	Got	
✓	`find_maximum(test_scores)`	5 45 86 95 76 28	`Maximum value is  95`	`Maximum value is  95`	✓

Passed all tests! ✓

Correct

Marks for this submission: 20.00/20.00.

Question **3**

Correct

Mark 20.00 out of 20.00

Write a Python Program to find minimum number of swaps required to sort an float array given by the user.

For example:

Test	Input	Result
minSwaps(arr)	5 2.3 6.5 4.1 9.5 7.5	2
minSwaps(arr)	6 3.2 1.4 5.6 9.2 4.5 6.2	4

Answer: (penalty regime: 0 %)

```
def minSwaps(arr):
    n=len(arr)
    swaps=0
    elements=[(arr[i],i) for i in range(n)]
    elements.sort(key=lambda x:x[0])
    visited=[False]*n
    for i in range(n):
        if visited[i] or elements[i][1]==i:
            continue
        cycle_size=0
        j=i
        while not visited[j]:
            visited[j]=True
            j=elements[j][1]
            cycle_size += 1
        swaps+=cycle_size-1
    return swaps


arr = []
n=int(input())
for i in range(n):
```

	Test	Input	Expected	Got	
✓	minSwaps(arr)	5 2.3 6.5 4.1 9.5 7.5	2	2	✓

	Test	Input	Expected	Got	
✓	`minSwaps(arr)`	6 3.2 1.4 5.6 9.2 4.5 6.2	4	4	✓
✓	`minSwaps(arr)`	4 2.3 6.1 4.2 3.1	1	1	✓

Passed all tests! ✓

Correct

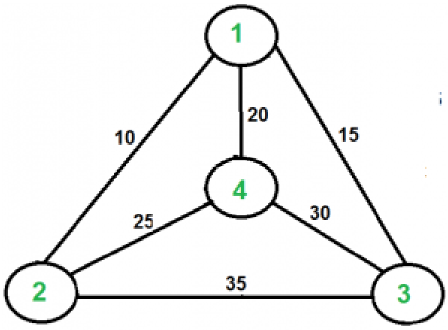
Marks for this submission: 20.00/20.00.

Question **4**

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):
    vertex =[]
    for i in range(V):
        if i !=s:
            vertex.append(i)
    min_path = maxsize
    next_permutation = permutations(vertex)
    for i in next_permutation:
        current_pathweight = 0
        k = s
        for j in i:
            current_pathweight += graph[k][j]
            k = j
        current_pathweight += graph[k][s]
        min_path = min(min_path, current_pathweight)

    return min_path
```

	Expected	Got	
✓	80	80	✓

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 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

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

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, v
```

	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	✓

	Test	Input	Expected	Got	
✓	`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.