Started on	Tuesday, 27 May 2025, 1:20 PM
State	Finished
Completed on	Tuesday, 27 May 2025, 2:18 PM
Time taken	58 mins 32 secs
Grade	**80.00** out of 100.00

Question **1**

Correct

Mark 20.00 out of 20.00

Write a python program to implement merge sort using iterative approach on the given list of values.

For example:

Test	Input	Result
Merge_Sort(S)	6 4 2 3 1 6 5	The Original array is: [4, 2, 3, 1, 6, 5] Array after sorting is: [1, 2, 3, 4, 5, 6]
Merge_Sort(S)	5 2 6 4 3 1	The Original array is: [2, 6, 4, 3, 1] Array after sorting is: [1, 2, 3, 4, 6]

Answer: (penalty regime: 0 %)

```
def Merge_Sort(S):
    s=len(S)
    if(s>1):
        m=s//2
        l=S[:m]
        r=S[m:]
        Merge_Sort(l)
        Merge_Sort(r)
        i=0
        j=0
        k=0

        l_s=len(l)
        r_s=len(r)
        while(i<l_s and j<r_s):
            if(l[i]<r[j]):
                S[k]=l[i]
                i+=1
            else:
                S[k]=r[j]
                j+=1
            k+=1
```

	Test	Input	Expected	Got	
✔	Merge_Sort(S)	6 4 2 3 1 6 5	The Original array is: [4, 2, 3, 1, 6, 5] Array after sorting is: [1, 2, 3, 4, 5, 6]	The Original array is: [4, 2, 3, 1, 6, 5] Array after sorting is: [1, 2, 3, 4, 5, 6]	✔

	Test	Input	Expected	Got	
✔	Merge_Sort(S)	5 2 6 4 3 1	The Original array is: [2, 6, 4, 3, 1] Array after sorting is: [1, 2, 3, 4, 6]	The Original array is: [2, 6, 4, 3, 1] Array after sorting is: [1, 2, 3, 4, 6]	✔
✔	Merge_Sort(S)	4 3 5 6 1	The Original array is: [3, 5, 6, 1] Array after sorting is: [1, 3, 5, 6]	The Original array is: [3, 5, 6, 1] Array after sorting is: [1, 3, 5, 6]	✔

Passed all tests! ✔

Correct

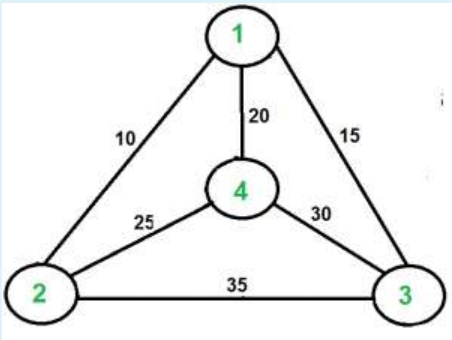
Marks for this submission: 20.00/20.00.

Question **2**

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 **3**

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

```
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 **4**

Incorrect

Mark 0.00 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):
################ Add your code here ##############

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))
```

Syntax Error(s)

```
Sorry: IndentationError: expected an indented block (__tester__.python3, line 4)
```

Incorrect

Marks for this submission: 0.00/20.00.

Question **5**

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 = [[-1] * (n + 1) for _ in range(n + 1)]
        dp[1][1] = grid[0][0]
        for m in range(1, (n << 1) - 1):
            for i in range(min(m, n - 1), max(-1, m - n), -1):
                for p in range(i, max(-1, m - n), -1):
                    j, q = m - i, m - p
                    if grid[i][j] == -1 or grid[p][q] == -1:
                        dp[i + 1][p + 1] = -1
                    else:
                        dp[i + 1][p + 1] = max(dp[i + 1][p + 1], dp[i][p + 1], dp[i + 1][p], dp[i][p]
                        if dp[i + 1][p + 1] != -1: dp[i + 1][p + 1] += grid[i][j] + (grid[p][q] if i
        return max(0, dp[-1][-1])
        n,m=len(grid),len(grid[0])
        dp = [[[-1 for i in range(m)] for j1 in range(n)] for j2 in range(n)]

        return f(0,0,m-1,dp)
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.