Started on	Wednesday, 28 May 2025, 9:21 AM
State	Finished
Completed on	Wednesday, 28 May 2025, 9:50 AM
Time taken	28 mins 46 secs
Grade	**80.00** out of 100.00

Question **1**

Not answered

Mark 0.00 out of 20.00

Given a string s, return *the longest palindromic substring* in s.

Example 1:

```
Input: s = "babad"
Output: "bab"
Explanation: "aba" is also a valid answer.
```

Example 2:

```
Input: s = "cbbd"
Output: "bb"
```

For example:

Test	Input	Result
ob1.longestPalindrome(str1)	ABCBCB	BCBCB

Answer: (penalty regime: 0 %)

Reset answer

```
1 class Solution(object):
2     def longestPalindrome(self, s):
3         ############  Add your code here ##############
4 ob1 = Solution()
5 str1=input()
6 print(ob1.longestPalindrome(str1))
```

Syntax Error(s)

```
  File "__tester__.python3", line 2
    You have two robots that can collect cherries for you.
        ^
SyntaxError: invalid syntax
```

Incorrect

Marks for this submission: 0.00/20.00.

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

```
def find_maximum(lst):
    maxi=lst[0]
    for i in lst:
        if i>maxi:
            maxi=i
    return maxi


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	✔
✔	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

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 n==0 or W==0:
        return 0
    if wt[n-1]>W:
        return knapSack(W, wt, val, n-1)
    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, 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)
        ### add code here
        dp=[[[-1]*n for _ in range(n)] for _ in range(n)]
        def f(x1,y1,x2):
            y2=x1+y1-x2
            if x1<0 or y1<0 or x2<0 or y2<0 or grid[x1][y1]==-1 or grid[x2][y2]==-1:
                return float('-inf')
            if x1==0 and y1==0 and x2==0 and y2==0:
                return grid[0][0]
            if dp[x1][y1][x2]!=-1:
                return dp[x1][y1][x2]
            cherries=grid[x1][y1]
            if x1!=x2 or y1!=y2:
                cherries+=grid[x2][y2]
            cherries+=max(
                        f(x1-1,y1,x2-1),
                         f(x1,y1-1,x2-1),
                         f(x1-1,y1,x2),
                         f(x1,y1-1,x2))
            dp[x1][y1][x2]=cherries
```

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

Passed all tests! ✔

Correct

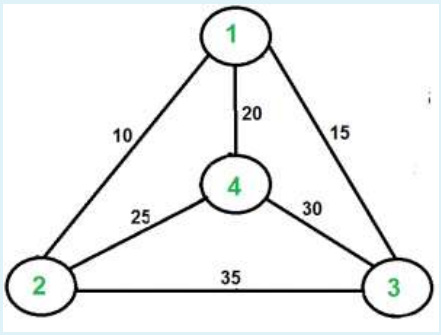
Marks for this submission: 20.00/20.00.

Question **5**

Correct

Mark 20.00 out of 20.00

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):

   #Write your code
    v=[]
    for i in range(V):
        if i!=s:
            v.append(i)
    mp=maxsize
    np=permutations(v)
    for i in np:
        k=s
        cp=0
        for j in i:
            cp+=graph[k][j]
            k=j
        cp+=graph[k][s]
        mp=min(cp,mp)
```

	Expected	Got	
✔	80	80	✔

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.