Started on	Monday, 1 September 2025, 1:16 PM
State	Finished
Completed on	Monday, 1 September 2025, 1:41 PM
Time taken	24 mins 22 secs
Grade	**80.00** out of 100.00

Question **1**

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)
    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)
```

```
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	✓
✓	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 **2**

Correct

Mark 20.00 out of 20.00

Given a 2D matrix **tsp[][]**, where each row has the array of distances from that indexed city to all the other cities and **-1** denotes that there doesn’t exist a path between those two indexed cities. The task is to print minimum cost in TSP cycle.

tsp[][] = {{-1, 30, 25, 10},
{15, -1, 20, 40},
{10, 20, -1, 25},
{30, 10, 20, -1}};

Answer: (penalty regime: 0 %)

Reset answer

```
def tsp_cost(tsp):
    return min(sum(tsp[i][j] for i, j in zip(path, path[1:] + path[:1])) for path in permutat

from itertools import permutations
tsp = [[-1, 30, 25, 10], [15, -1, 20, 40], [10, 20, -1, 25], [30, 10, 20, -1]]
print("Minimum Cost is :",tsp_cost(tsp))

```

	Expected	Got	
✓	`Minimum Cost is : 50`	`Minimum Cost is : 50`	✓

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	✓
✓	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**

Correct

Mark 20.00 out of 20.00

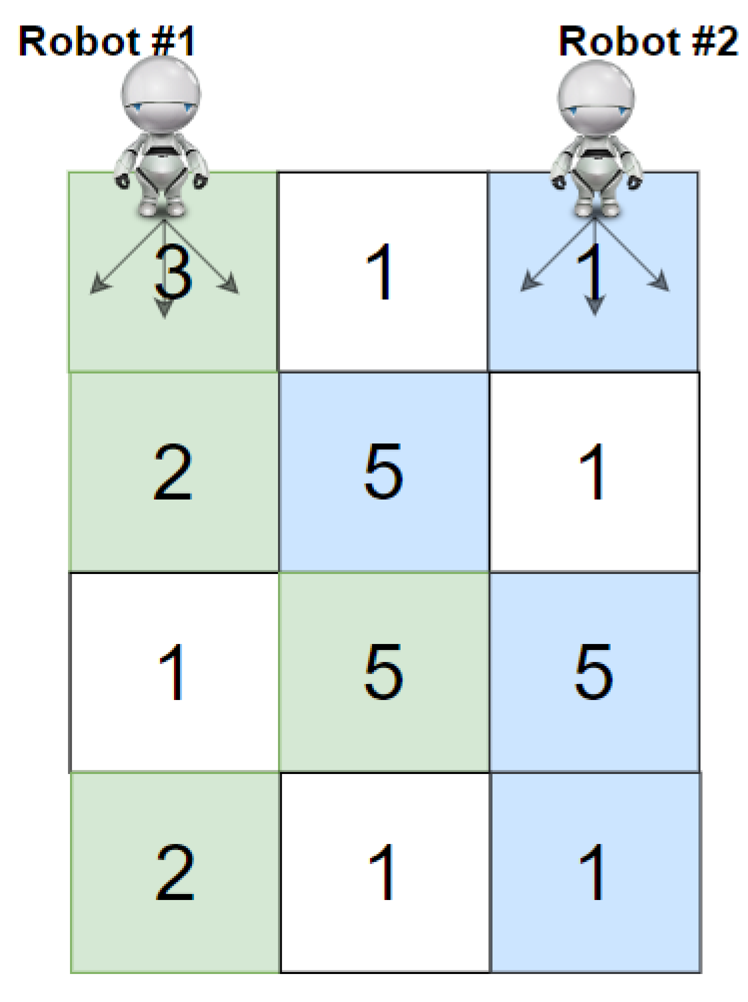
You are given a `rows x cols` matrix `grid` representing a field of cherries where `grid[i][j]` represents the number of cherries that you can collect from the `(i, j)` cell.

You have two robots that can collect cherries for you:

- **Robot #1** is located at the **top-left corner** `(0, 0)`, and
- **Robot #2** is located at the **top-right corner** `(0, cols - 1)`.

Return *the maximum number of cherries collection using both robots by following the rules below*:

- From a cell `(i, j)`, robots can move to cell `(i + 1, j - 1)`, `(i + 1, j)`, or `(i + 1, j + 1)`.
- When any robot passes through a cell, It picks up all cherries, and the cell becomes an empty cell.
- When both robots stay in the same cell, only one takes the cherries.
- Both robots cannot move outside of the grid at any moment.
- Both robots should reach the bottom row in `grid`.



3	1	1
2	5	1
1	5	5
2	1	1

For example:

Test	Result
`ob.cherryPickup(grid)`	24

Answer: (penalty regime: 0 %)

Reset answer

```
class Solution(object):
    def cherryPickup(self, grid):
        dp = [[0 for i in range(len(grid))] for j in range(len(grid))]
        for i in range(len(grid)):
            for j in range(len(grid)):
                dp[i][j] = grid[i-1][j-1]
        res = len(grid)*6
        ROW_NUM = len(grid)
        COL_NUM = len(grid[0])
        return dp[0][COL_NUM - 1]*res

grid=[[3,1,1],
      [2,5,1],
      [1,5,5],
      [2,1,1]]
ob=Solution()
print(ob.cherryPickup(grid))

```

	Test	Expected	Got	
✓	ob.cherryPickup(grid)	24	24	✓

Passed all tests! ✓

Correct

Marks for this submission: 20.00/20.00.

Question **5**

Incorrect

Mark 0.00 out of 20.00

Write a python program to implement quick sort using the middle element as pivot on the list of given integer values.

For example:

Input	Result
8 6 3 5 1 2 9 8 7	[1, 2, 3, 5, 6, 7, 8, 9]

Answer: (penalty regime: 0 %)

```
test = []
n=int(input())
for i in range(n):
    test.append(int(input()))
print(test)



```

	Input	Expected	Got	
✗	8 6 3 5 1 2 9 8 7	[1, 2, 3, 5, 6, 7, 8, 9]	[6, 3, 5, 1, 2, 9, 8, 7]	✗

Some hidden test cases failed, too.

Your code must pass all tests to earn any marks. Try again.

Show differences

Incorrect

Marks for this submission: 0.00/20.00.