Started on	Tuesday, 20 May 2025, 2:50 PM
State	Finished
Completed on	Tuesday, 20 May 2025, 3:41 PM
Time taken	50 mins 44 secs
Grade	**80.00** out of 100.00

Question **1**

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):
    if len(lst)==0:
        return 0
    max_=lst[0]
    for i in lst:
        if 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 **2**

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):
    dp=[[0]*(W+1) for _ in range(n+1)]
    for i in range(n+1):
        dp[i][0]=0
    for j in range(W+1):
        dp[0][j]=0
    for i in range(n+1):
        for j in range(W+1):
            if j<wt[i-1]:
                dp[i][j]=dp[i-1][j]
            else:
                dp[i][j]=max(dp[i-1][j],dp[i-1][j-wt[i-1]]+val[i-1])
    return dp[n][W]

x=int(input())
y=int(input())
W=int(input())
val=[]
wt=[]
for i in range(x):
    val.append(int(input()))
```

	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 **3**

Incorrect

Mark 0.00 out of 20.00

Write a python program to check whether Hamiltonian path exits in the given graph.

For example:

Test	Result
`Hamiltonian_path(adj, N)`	`YES`

Answer: (penalty regime: 0 %)

Reset answer

```
 1 def Hamiltonian_path(adj, N):
 2     ######################### Add your Code here ##########################
 3 adj = [ [ 0, 1, 1, 1, 0 ] ,
 4         [ 1, 0, 1, 0, 1 ],
 5         [ 1, 1, 0, 1, 1 ],
 6         [ 1, 0, 1, 0, 0 ] ]
 7 
 8 N = len(adj)
 9 
10 if (Hamiltonian_path(adj, N)):
11     print("YES")
12 else:
13     print("NO")
```

Syntax Error(s)

```
Sorry: IndentationError: expected an indented block (__tester__.python3, line 3)
```

Incorrect

Marks for this submission: 0.00/20.00.

Question **4**

Correct

Mark 20.00 out of 20.00

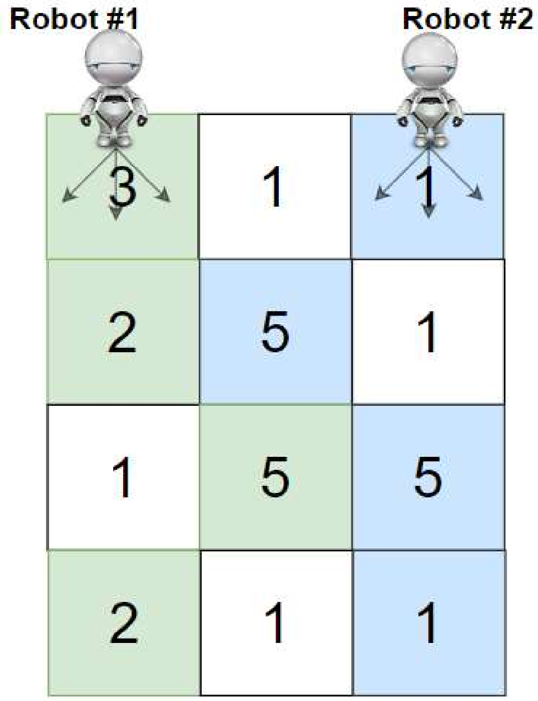
You are given a `rows x cols` matrix `grid` representing a field of cherries where `grid[i][j]` represents the number of cherries that you can collect from the `(i, j)` cell.

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

```python
class Solution(object):
    def cherryPickup(self, grid):
        ROW_NUM = len(grid)
        COL_NUM = len(grid[0])
        dp = [[[float('-inf')] * COL_NUM for _ in range(COL_NUM)] for _ in range(
        dp[0][0][COL_NUM - 1] = grid[0][0] + grid[0][COL_NUM - 1]
        for i in range(1, ROW_NUM):
            for j1 in range(COL_NUM):
                for j2 in range(COL_NUM):
                    curr_cherries = grid[i][j1]
                    if j1 != j2:
                        curr_cherries += grid[i][j2]
                    for prev_j1 in range(j1 - 1, j1 + 2):
                        for prev_j2 in range(j2 - 1, j2 + 2):
                            if 0 <= prev_j1 < COL_NUM and 0 <= prev_j2 < COL_NUM:
                                prev_cherries = dp[i - 1][prev_j1][prev_j2]
                                dp[i][j1][j2] = max(dp[i][j1][j2], curr_cherries
                        dp[i][j1][j2] = max(dp[i][j1][j2], curr_cherries

        return max(0, dp[ROW_NUM - 1][0][COL_NUM - 1])

grid=[[3,1,1],
```

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

Passed all tests! ✓

Correct

Marks for this submission: 20.00/20.00.

Question **5**

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

```
from typing import DefaultDict


INT_MAX = 2147483647


def findMinRoute(tsp):
    sum = 0
    counter = 0
    j = 0
    i = 0
    min = INT_MAX
    visitedRouteList = DefaultDict(int)


    visitedRouteList[0] = 1
    route = [0] * len(tsp)


    while i < len(tsp) and j < len(tsp[i]):
        if counter >= len(tsp[i]) - 1:
            break
```

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

Passed all tests! ✓

Correct

Marks for this submission: 20.00/20.00.