Started on	Thursday, 8 May 2025, 1:44 PM
State	Finished
Completed on	Thursday, 8 May 2025, 5:31 PM
Time taken	3 hours 46 mins
Overdue	1 hour 46 mins
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

Ace editor not ready. Perhaps reload page?
Falling back to raw text area.

```
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 **2**

Correct

Mark 20.00 out of 20.00

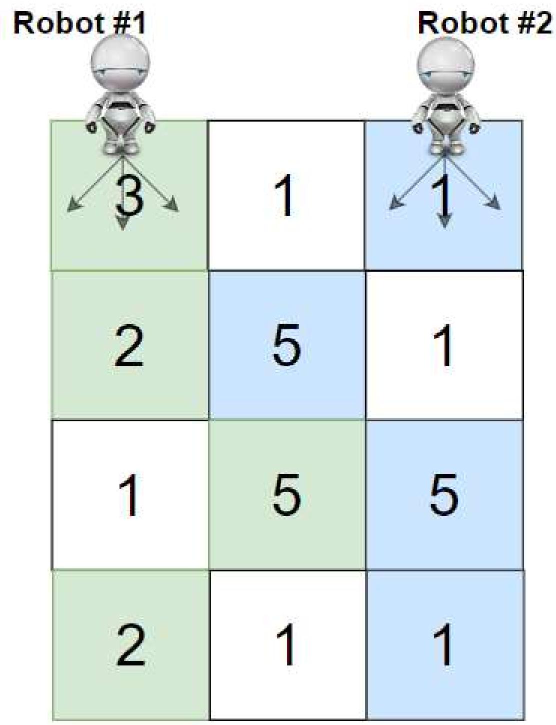
You are given a `rows x cols` matrix `grid` representing a field of cherries where `grid[i][j]` represents the number of cherries that you can collect from the `(i, j)` cell.

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

Ace editor not ready. Perhaps reload page?
Falling back to raw text area.

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

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

Passed all tests! ✓

Correct

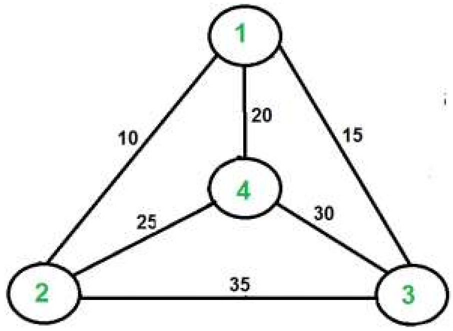
Marks for this submission: 20.00/20.00.

Question **3**

Correct

Mark 20.00 out of 20.00

Solve Travelling Sales man Problem for the following graph



Answer: (penalty regime: 0 %)

Reset answer

Ace editor not ready. Perhaps reload page?
Falling back to raw text area.

```
from sys import maxsize
from itertools import permutations
V = 4


def travellingSalesmanProblem(graph, s):
    vertices=[]
    for i in range(V):
        if i!=s:
            vertices.append(i)
    min_path=maxsize
    next_permutations=permutations(vertices)
    for i in next_permutations:
        current_weight=0
        k=s
        for j in i:
            current_weight+=graph[k][j]
            k=j
```

	Expected	Got	
✓	80	80	✓

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 using dynamic programming for 0/1 knapsack problem.

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

Ace editor not ready. Perhaps reload page?
Falling back to raw text area.

```
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()))
```

	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 40 50 90 110 10 20 30	`The maximum value that can be put in a knapsack of capacity W is:  160`	`The maximum value that can be put in a knapsack of capacity W is:  160`	✓

Passed all tests! ✓

Correct

Marks for this submission: 20.00/20.00.

Question **5**

Incorrect

Mark 0.00 out of 20.00

SUBSET SUM PROBLEM

Given a set of positive integers, and a value sum, determine that the sum of the subset of a given set is equal to the given sum.

Write the program for subset sum problem.

INPUT

1.no of elements

2.Input the given elements

3.Get the target sum

OUTPUT

True , if subset with required sum is found

False , if subset with required sum is not found

For example:

Input	Result
5 4 16 5 23 12 9	4 16 5 23 12 True,subset found

Answer: (penalty regime: 0 %)

Reset answer

Ace editor not ready. Perhaps reload page?
Falling back to raw text area.

```
def SubsetSum(a,i,sum,target,n):

# Write your code here










a=[]
size=int(input())
for i in range(size):
    x=int(input())
    a.append(x)

target=int(input())
```

Syntax Error(s)

```
Sorry: IndentationError: expected an indented block (__tester__.python3, line 12)
```

Incorrect

Marks for this submission: 0.00/20.00.