Started on	Monday, 19 May 2025, 3:30 PM
State	Finished
Completed on	Monday, 19 May 2025, 3:34 PM
Time taken	3 mins 41 secs
Grade	**80.00** out of 100.00

Question **1**

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

```
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()))
for y in range(y):
```

	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 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 **2**

Correct

Mark 20.00 out of 20.00

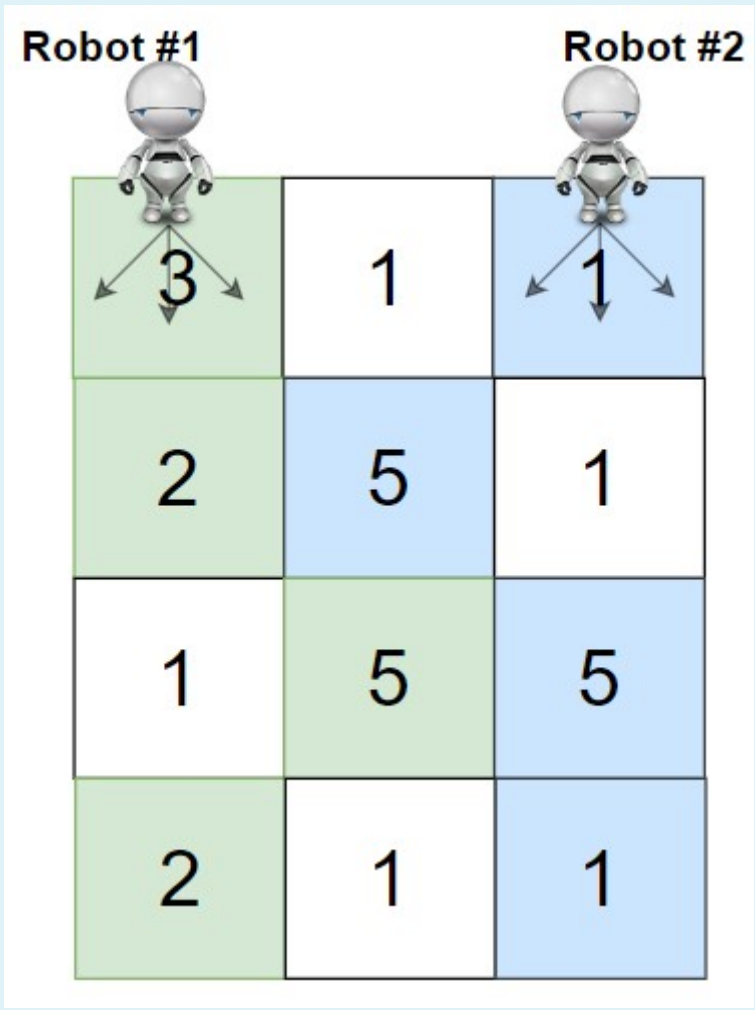
You are given a `rows x cols` matrix `grid` representing a field of cherries where `grid[i][j]` represents the number of cherries that you can collect from the `(i, j)` cell.

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`.



For example:

Test	Result
`ob.cherryPickup(grid)`	`24`

Answer: (penalty regime: 0 %)

Reset answer

```
from itertools import product
class Solution(object):
    def cherryPickup(self, grid):
        ROW_NUM = len(grid)
        COL_NUM = len(grid[0])
        dp=[[0]*COL_NUM for _ in range(COL_NUM)]
        for r in reversed(range(ROW_NUM)):
            cur_dp=[[0]*COL_NUM for _ in range(COL_NUM)]

            for c1 in range(COL_NUM-1):
                for c2 in range(c1+1,COL_NUM):
                    max_cherry=0
                    cherry=grid[r][c1]+grid[r][c2]

                    for c1d,c2d in product([-1,0,1],[-1,0,1]):
                        nc1,nc2=c1+c1d,c2+c2d

                        if nc1<0 and nc2==COL_NUM:
                            continue
                        max_cherry=max(max_cherry,cherry+dp[nc1-1][nc2-1])

                    cur_dp[c1][c2]=max_cherry
```

	Test	Expected	Got	
✔	`ob.cherryPickup(grid)`	24	24	✔

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 sort the first half of the list using merge sort with float values.

For example:

Input	Result
5 6.3 5.2 4.1 1.5 9.8	Given array is 6.3 5.2 4.1 1.5 9.8 Sorted array is 1.5 6.3 5.2 4.1 9.8

Answer: (penalty regime: 0 %)

```
Write a recursive python function to perform merge sort on the unsorted list of float values.

def mergesort(li):
    arr=li
    if len(arr)>1:
        mid=len(arr)//2
        l=arr[:mid]
        r=arr[mid:]
        mergesort(l)
        mergesort(r)
        merge(l,r,arr)
def merge(l,r,arr):
    i=j=k=0
    while i<len(l) and j<len(r):
        if l[i]<r[j]:
            arr[k]=l[i]
            i+=1
            k+=1
        else:
            arr[k]=r[j]
            j+=1
            k+=1
```

Syntax Error(s)

```
  File "__tester__.python3", line 1
    Write a recursive python function to perform merge sort on the unsorted list of float values.
          ^
SyntaxError: invalid syntax
```

Incorrect

Marks for this submission: 0.00/20.00.

Question **4**

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_t=lst[0]
    for i in lst:
        if i>max_t:
            max_t=i
    return max_t

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

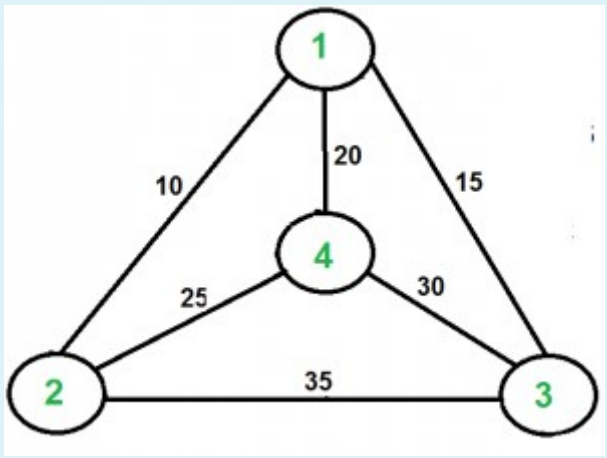
Marks for this submission: 20.00/20.00.

Question **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):

    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.