

Started on	Wednesday, 10 September 2025, 8:52 AM
State	Finished
Completed on	Wednesday, 10 September 2025, 9:58 AM
Time taken	1 hour 5 mins
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

```

1 def knapSack(W, wt, val, n):
2     if n==0 or W==0:
3         return 0
4     if wt[n-1]>W:
5         return knapSack(W, wt, val, n-1)
6     return max(val[n-1]+knapSack(W-wt[n-1], wt, val, n-1),knapSack(W, wt, val, n-1))
7
8 x=int(input())
9 y=int(input())
10 W=int(input())
11 val=[]
12 wt=[]
13 for i in range(x):
14     val.append(int(input()))
15 for y in range(y):
16     wt.append(int(input()))
17 n = len(val)
18 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! ✓



Marks for this submission: 20.00/20.00.

Question 2

Correct

Mark 20.00 out of 20.00

Given an integer array `nums`, find the contiguous subarray (containing at least one number) which has the largest sum and return *its sum*.

A **subarray** is a **contiguous** part of an array.

Example 1:

Input: `nums = [-2,1,-3,4,-1,2,1,-5,4]`

Output: 6

Explanation: `[4,-1,2,1]` has the largest sum = 6.

For example:

Test	Input	Result
<code>s.maxSubArray(A)</code>	9 -2 1 -3 4 -1 2 1 -5 4	The sum of contiguous sublist with the largest sum is 6

Answer: (penalty regime: 0 %)

[Reset answer](#)

```

1 class Solution:
2     def maxSubArray(self,A):
3         ##### Add your Code here
4         max_sum = A[0]
5         current_sum = A[0]
6         for i in range(1, len(A)):
7             current_sum = max(A[i], current_sum + A[i])
8             max_sum = max(max_sum, current_sum)
9         return max_sum
10
11 A=[]
12 n=int(input())
13 for i in range(n):
14     A.append(int(input()))
15 s=Solution()
16 print("The sum of contiguous sublist with the largest sum is",s.maxSubArray(A))

```

	Test	Input	Expected	Got	
✓	<code>s.maxSubArray(A)</code>	9 -2 1 -3 4 -1 2 1 -5 4	The sum of contiguous sublist with the largest sum is 6	The sum of contiguous sublist with the largest sum is 6	✓

	Test	Input	Expected	Got	
✓	s.maxSubArray(A)	5 5 4 -1 7 8	The sum of contiguous sublist with the largest sum is 23	The sum of contiguous sublist with the largest sum is 23	✓

Passed all tests! ✓

Correct

Marks for this submission: 20.00/20.00.

Question 3

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

```
1 from sys import maxsize
2 from itertools import permutations
3 V = 4
4
5
6 def travellingSalesmanProblem(graph, s):
7
8     #Write your code
9     v=[]
10    for i in range(V):
11        if i!=s:
12            v.append(i)
13    mp=maxsize
14    np=permutations(v)
15    for i in np:
16        k=s
17        cp=0
18        for j in i:
19            cp+=tsp[k][j]
20            k=j
21        cp+=tsp[k][s]
22        mp=min(cp,mp)
```

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

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 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

```

1 def find_maximum(lst):
2     maxi=lst[0]
3     for i in lst:
4         if i>maxi:
5             maxi=i
6     return maxi
7
8
9 test_scores = []
10 n=int(input())
11 for i in range(n):
12     test_scores.append(int(input()))
13 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! ✓



Marks for this submission: 20.00/20.00.

Question 5

Incorrect

Mark 0.00 out of 20.00

Create a python program to for the following problem statement.

You are given an $n \times n$ 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

```

1 class Solution:
2     def cherryPickup(self, grid):
3         n = len(grid)
4
5         return f(0,0,m-1,dp)
6 obj=Solution()
7 grid=[[0,1,-1],[1,0,-1],[1,1,1]]
8 print(obj.cherryPickup(grid))

```

	Test	Expected	Got	
✖	obj.cherryPickup(grid)	5	***Run error*** Traceback (most recent call last): File "__tester__.python3", line 8, in <module> print(obj.cherryPickup(grid)) File "__tester__.python3", line 5, in cherryPickup return f(0,0,m-1,dp) NameError: name 'f' is not defined	✖

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

Show differences

Incorrect

Marks for this submission: 0.00/20.00.