Started on	Saturday, 17 May 2025, 11:12 AM
State	Finished
Completed on	Sunday, 18 May 2025, 11:37 PM
Time taken	1 day 12 hours
Overdue	1 day 10 hours
Grade	**100.00** out of 100.00

Question 1

Correct

Mark 20.00 out of 20.00

Create a Python Function to find the total number of distinct ways to get a change of 'target' from an unlimited supply of coins in set 'S'.

For example:

Test	Input	Result
count(S, len(S) - 1, target)	3 4 1 2 3	The total number of ways to get the desired change is 4

Answer: (penalty regime: 0 %)

Reset answer

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

```
def count(S, n, target):
    ###################  Add Your Code Here #################
    #Start here
    if target == 0:
        return 1
    if target < 0 or n < 0:
        return 0
    incl = count(S, n, target - S[n])
    excl = count(S, n - 1, target)
    return incl + excl
    #End here

if __name__ == '__main__':
    S = []#[1, 2, 3]
    n=int(input())
    target = int(input())
    for i in range(n):
        S.append(int(input()))
```

	Test	Input	Expected	Got	
✔	count(S, len(S) - 1, target)	3 4 1 2 3	The total number of ways to get the desired change is 4	The total number of ways to get the desired change is 4	✔
✔	count(S, len(S) - 1, target)	3 11 1 2 5	The total number of ways to get the desired change is 11	The total number of ways to get the desired change is 11	✔

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.

Question **2**

Correct

Mark 20.00 out of 20.00

Write a Python Program for printing Minimum Cost Simple Path between two given nodes in a directed and weighted graph

For example:

Test	Result
`minimumCostSimplePath(s, t, visited, graph)`	`-3`

Answer: (penalty regime: 0 %)

Reset answer

Ace editor not ready. Perhaps reload page?

Falling back to raw text area.

```
import sys
V = 5
INF = sys.maxsize
def minimumCostSimplePath(u, destination,
                          visited, graph):
########  Add your code here ###############
    #Start here
    if (u == destination):
        return 0
    visited[u] = 1
    ans = INF
    for i in range(V):
        if (graph[u][i] != INF and not visited[i]):
            curr = minimumCostSimplePath(i, destination,
                                         visited, graph)
            if (curr < INF):
                ans = min(ans, graph[u][i] + curr)
    visited[u] = 0
```

	Test	Expected	Got	
✔	`minimumCostSimplePath(s, t, visited, graph)`	`-3`	`-3`	✔

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 contiguous sub array using Dynamic Programming.

For example:

Test	Input	Result
maxSubArraySum(a,len(a))	8 -2 -3 4 -1 -2 1 5 -3	Maximum contiguous sum is 7

Answer: (penalty regime: 0 %)

Ace editor not ready. Perhaps reload page?

Falling back to raw text area.

```
def maxSubArraySum(a,size):
    ####################  Add your Code here #############
    #Start here
    max_so_far = a[0]
    max_ending_here = 0
    for i in range(0, size):
        max_ending_here = max_ending_here + a[i]
        if max_ending_here < 0:
            max_ending_here = 0
        elif (max_so_far < max_ending_here):
            max_so_far = max_ending_here

    return max_so_far
    #End here
n=int(input())
a =[] #[-2, -3, 4, -1, -2, 1, 5, -3]
for i in range(n):
    a.append(int(input()))
```

	Test	Input	Expected	Got	
✔	maxSubArraySum(a,len(a))	8 -2 -3 4 -1 -2 1 5 -3	Maximum contiguous sum is 7	Maximum contiguous sum is 7	✔
✔	maxSubArraySum(a,len(a))	5 1 2 3 -4 -6	Maximum contiguous sum is 6	Maximum contiguous sum is 6	✔

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 Minimum number of jumps to reach end of the array using naive method(recursion) using float values

For example:

Test	Input	Result
minJumps(arr, 0, n-1)	6 2.3 7.4 6.3 1.5 8.2 0.1	Minimum number of jumps to reach end is 2

Answer: (penalty regime: 0 %)

Reset answer

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

```
def minJumps(arr, l, h):
    ###########   Add your code here ###########
    #Start here
    if (h == l):
        return 0
    if (arr[l] == 0):
        return float('inf')
    min = float('inf')
    for i in range(l + 1, h + 1):
        if (i < l + arr[l] + 1):
            jumps = minJumps(arr, i, h)
            if (jumps != float('inf') and
                        jumps + 1 < min):
                min = jumps + 1

    return min
    #End here
arr = []
```

	Test	Input	Expected	Got	
✔	minJumps(arr, 0, n-1)	6 2.3 7.4 6.3 1.5 8.2 0.1	Minimum number of jumps to reach end is 2	Minimum number of jumps to reach end is 2	✔
✔	minJumps(arr, 0, n-1)	10 3.2 3.2 5 6.2 4.9 1.2 5.0 7.3 4.6 6.2	Minimum number of jumps to reach end is 2	Minimum number of jumps to reach end is 2	✔

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.

Question **5**

Correct

Mark 20.00 out of 20.00

GRAPH COLORING PROBLEM

Given an undirected graph and a number m, determine if the graph can be coloured with at most m colours such that no two adjacent vertices of the graph are colored with the same color. Here coloring of a graph means the assignment of colors to all vertices.

Input-Output format:

Input:

1. A 2D array graph[V][V] where V is the number of vertices in graph and graph[V][V] is an adjacency matrix representation of the graph. A value graph[i][j] is 1 if there is a direct edge from i to j, otherwise graph[i][j] is 0.
2. An integer m is the maximum number of colors that can be used.

Output:

An array color[V] that should have numbers from 1 to m. color[i] should represent the color assigned to the ith vertex.

Example:

```
Input:
graph = {0, 1, 1, 1},
        {1, 0, 1, 0},
        {1, 1, 0, 1},
        {1, 0, 1, 0}
Output:
Solution Exists:
Following are the assigned colors
 1  2  3  2
Explanation: By coloring the vertices
with following colors, adjacent
vertices does not have same colors
```

```
Input:
graph = {1, 1, 1, 1},
        {1, 1, 1, 1},
        {1, 1, 1, 1},
        {1, 1, 1, 1}
Output: Solution does not exist.
Explanation: No solution exits.
```

Answer: (penalty regime: 0 %)

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

```
class Graph:
    def __init__(self,vertices):
        self.V=vertices
        self.Graph=[[0 for column in range(vertices)]for row in range(vertices)]
    def isSafe(self,v,colour,c):
        for i in range(self.V):
            if self.graph[v][i]==1 and colour[i]==c:
                return False
        return True
    def graphColourUtil(self,m,colour,v):
        if v==self.V:
            return True
        for c in range(1,m+1):
            if self.isSafe(v,colour,c):
                colour[v]=c
                if self.graphColourUtil(m,colour,v+1):
                    return True
                colour[v]=c
```

	Test	Expected	Got	
✔	g = Graph(4) g.graph = [[0, 1, 1, 1], [1, 0, 1, 0], [1, 1, 0, 1], [1, 0, 1, 0]] m = 3 g.graphColouring(m)	Solution exist and Following are the assigned colours: 1 2 3 2	Solution exist and Following are the assigned colours: 1 2 3 2	✔

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.