Started on	Saturday, 3 May 2025, 8:18 AM
State	Finished
Completed on	Saturday, 3 May 2025, 8:48 AM
Time taken	30 mins 6 secs
Grade	**80.00** 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

```
def count(S, n, target):
    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

if __name__ == '__main__':
    S = []#[1, 2, 3]
    n=int(input())
    target = int(input())
    for i in range(n):
        S.append(int(input()))
    print('The total number of ways to get the desired change is',
        count(S, len(S) - 1, target))
```

	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**

Incorrect

Mark 0.00 out of 20.00

Write a python program to implement knight tour problem using backtracking

For example:

Input	Result
5	Found a solution 01 20 11 14 03 10 15 02 19 12 21 24 13 04 07 16 09 06 23 18 25 22 17 08 05

Answer: (penalty regime: 0 %)

Reset answer

```
BOARD_SIZE = int(input())
board = [[0 for i in range(BOARD_SIZE)] for j in range(BOARD_SIZE)]
STEPS = [[-1, 2], [1, 2], [-2, 1], [2, 1], [1, -2], [-1, -2], [2, -1], [-2, -1]]


def solve_knights_tour(x, y, step_count):
    ########################## Add your code here ############################3

def is_safe(x, y):
    return 0 <= x < BOARD_SIZE and 0 <= y < BOARD_SIZE and board[x][y] == 0


def print_solution():
    for row in board:
        for col in row:
            print("0" + str(col) if col < 10 else col, end=" ")
        print()


board[0][0] = 1     # First move is at (0, 0)

if solve_knights_tour(0, 0, 2):
```

Syntax Error(s)

```
Sorry: IndentationError: expected an indented block (__tester__.python3, line 9)
```

Incorrect

Marks for this submission: 0.00/20.00.

Question **3**

Correct

Mark 20.00 out of 20.00

Write a Python program to Implement Minimum cost path in a Directed Graph

For example:

Test	Result
getMinPathSum(graph, visited, necessary, source, dest, 0);	12

Answer: (penalty regime: 0 %)

Reset answer

```
minSum = 1000000000
def getMinPathSum(graph, visited, necessary,src, dest, currSum):
    global minSum
    if src==dest:
        flag=True
        for i in necessary:
            if not visited[i]:
                flag=False
                break
        if flag:
            minSum=min(minSum,currSum)
        return
    for node in graph[src]:
        if not visited[node[0]]:
            visited[node[0]]=True
            getMinPathSum(graph,visited,necessary,node[0],dest,currSum+node[1])
            visited[node[0]]=False
    visited[src]=False
if __name__=='__main__':
    graph=dict()
    graph[0] = [ [ 1, 2 ], [ 2, 3 ], [ 3, 2 ] ];
    graph[1] = [ [ 4, 4 ], [ 0, 1 ] ];
```

	Test	Expected	Got	
✔	getMinPathSum(graph, visited, necessary, source, dest, 0);	12	12	✔

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.

Question **4**

Correct

Mark 20.00 out of 20.00

Write a python program to find the maximum contiguous subarray on the given float array using kadane's algorithm.

For example:

Test	Input	Result
s.maxSubArray(A)	5 -9.6 -3.5 6.3 8.31 9.2	The sum of contiguous sublist with the largest sum is 23.8

Answer: (penalty regime: 0 %)

Reset answer

```
class Solution:
    def maxSubArray(self,A):
        res=0
        mm= -10000
        for v in A:
            res+=v
            mm=max(mm,res)
            if res<0:
                res=0
        return mm
A =[]
n=int(input())
for i in range(n):
    A.append(float(input()))
s=Solution()
print("The sum of contiguous sublist with the largest sum is {:.1f}".format(s.maxSubArray(A)))
```

	Test	Input	Expected	Got	
✔	s.maxSubArray(A)	5 -9.6 -3.5 6.3 8.31 9.2	The sum of contiguous sublist with the largest sum is 23.8	The sum of contiguous sublist with the largest sum is 23.8	✔
✔	s.maxSubArray(A)	7 2.3 6.5 4.6 -7.8 -2.8 -1.6 9.8	The sum of contiguous sublist with the largest sum is 13.4	The sum of contiguous sublist with the largest sum is 13.4	✔

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.

Question **5**

Correct

Mark 20.00 out of 20.00

Print All Paths With Minimum Jumps

```
1. You are given a number N representing number of elements.
2. You are given N space separated numbers (ELE : elements).
3. Your task is to find & print
    3.1) "MINIMUM JUMPS" need from 0th step to (n-1)th step.
    3.2) all configurations of "MINIMUM JUMPS".
NOTE: Checkout sample question/solution video inorder to have more insight.
```

For example:

Test	Input	Result
minJumps(arr)	10 3 3 0 2 1 2 4 2 0 0	0 -> 3 -> 5 -> 6 -> 9 0 -> 3 -> 5 -> 7 -> 9

Answer: (penalty regime: 0 %)

Reset answer

```
from queue import Queue
import sys
class Pair(object):
    idx = 0
    psf = ""
    jmps = 0
    def __init__(self, idx, psf, jmps):
        self.idx = idx
        self.psf = psf
        self.jmps = jmps
def minJumps(arr):
    MAX_VALUE = sys.maxsize
    dp = [MAX_VALUE for i in range(len(arr))]
    n = len(dp)
    dp[n - 1] = 0
    for i in range(n - 2, -1, -1):
        steps = arr[i]
        minimum = MAX_VALUE
        for j in range(1, steps + 1, 1):
            if i + j >= n:
                break
            if ((dp[i + j] != MAX_VALUE) and
```

	Test	Input	Expected	Got	
✔	minJumps(arr)	10 3 3 0 2 1 2 4 2 0 0	0 -> 3 -> 5 -> 6 -> 9 0 -> 3 -> 5 -> 7 -> 9	0 -> 3 -> 5 -> 6 -> 9 0 -> 3 -> 5 -> 7 -> 9	✔
✔	minJumps(arr)	7 5 5 0 3 2 3 6	0 -> 1 -> 6 0 -> 3 -> 6 0 -> 4 -> 6 0 -> 5 -> 6	0 -> 1 -> 6 0 -> 3 -> 6 0 -> 4 -> 6 0 -> 5 -> 6	✔

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.