Started on	Saturday, 15 March 2025, 2:16 PM
State	Finished
Completed on	Saturday, 15 March 2025, 2:42 PM
Time taken	26 mins 43 secs
Grade	**100.00** out of 100.00

Question **1**

Correct

Mark 20.00 out of 20.00

Write a Python program using A Naive recursive implementation of Minimum Cost Path Problem.

For example:

Input	Result
3 3	8

Answer: (penalty regime: 0 %)

Reset answer

```
R = int(input())
C = int(input())
def minCost(cost, m, n):
    tc = [[0 for x in range(C)] for x in range(R)]
    tc[0][0] = cost[0][0]
    for i in range(1, m+1):
        tc[i][0] = tc[i-1][0] + cost[i][0]
    for j in range(1, n+1):
        tc[0][j] = tc[0][j-1] + cost[0][j]
    for i in range(1, m+1):
        for j in range(1, n+1):
            tc[i][j] = min(tc[i-1][j-1], tc[i-1][j], tc[i][j-1]) + cost[i][j]

    return tc[m][n]

cost = [[1, 2, 3],
        [4, 8, 2],
        [1, 5, 3]]
print(minCost(cost, R-1, C-1))

```

	Input	Expected	Got	
✔	3 3	8	8	✔

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.

Question **2**

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 %)

```
def maxSubArraySum(a,size):
    maxs =a[0]
    curr = a[0]
    for i in range(1,size):
        curr = max(a[i], curr+ a[i])
        maxs = max(maxs,curr)
    return maxs

n=int(input())
a =[]
for i in range(n):
    a.append(int(input()))
print("Maximum contiguous sum is", maxSubArraySum(a,n))
```

	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 **3**

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

```
def minJumps(arr, l, h):
    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


arr = []
n = int(input())
for i in range(n):
    arr.append(float(input()))
print('Minimum number of jumps to reach','end is', minJumps(arr, 0, n-1))

```

	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	✔

Question **4**

Correct

Mark 20.00 out of 20.00

Write a python program to implement quick sort using last element as pivot on the given list of integers.

For example:

Test	Input	Result
quickSort(arr,0,n-1)	6 21 54 30 12 10 6	Sorted array is: 6 10 12 21 30 54

Answer: (penalty regime: 0 %)

```
def partition(arr,l, r):
    pivot = arr[r]
    ptr = l - 1
    for i in range(l, r):
        if arr[i] <= pivot:
            ptr += 1
            arr[ptr], arr[i] = arr[i], arr[ptr]
    arr[ptr + 1], arr[r] = arr[r], arr[ptr + 1]
    return ptr + 1
def quickSort(arr,l, r):
    if l < r:
        pi = partition( arr,l,r)
        quickSort( arr,l,pi-1)
        quickSort(arr,pi+1,r)
    return arr

n = int(input())
arr = []

for _ in range(n):
    num = int(input())
    arr.append(num)
```

	Test	Input	Expected	Got	
✔	quickSort(arr,0,n-1)	6 21 54 30 12 10 6	Sorted array is: 6 10 12 21 30 54	Sorted array is: 6 10 12 21 30 54	✔
✔	quickSort(arr,0,n-1)	5 41 21 30 12 98	Sorted array is: 12 21 30 41 98	Sorted array is: 12 21 30 41 98	✔
✔	quickSort(arr,0,n-1)	8 2 6 7 4 9 3 1 5	Sorted array is: 1 2 3 4 5 6 7 9	Sorted array is: 1 2 3 4 5 6 7 9	✔

Question **5**

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.