Started on	Thursday, 24 April 2025, 2:22 PM
State	Finished
Completed on	Monday, 28 April 2025, 10:38 AM
Time taken	3 days 20 hours
Overdue	3 days 18 hours
Grade	**80.00** out of 100.00

Question **1**

Correct

Mark 20.00 out of 20.00

Create a python function to compute the fewest number of coins that we need to make up the amount given.

For example:

Test	Input	Result
ob1.coinChange(s,amt)	3 11 1 2 5	3

Answer: (penalty regime: 0 %)

Reset answer

```
class Solution(object):
    def coinChange(self, coins, amount):
        ####################      Add your Code Here ###########
        #End here
        if amount == 0 :
            return 0
        if min(coins) > amount:
            return -1
        dp = [-1 for i in range(0, amount + 1)]
        for i in coins:
            if i > len(dp) - 1:
                continue
            dp[i] = 1
            for j in range(i + 1, amount + 1):
                if dp[j - i] == -1:
                    continue
                elif dp[j] == -1:
                    dp[j] = dp[j - i] + 1
                else:
                    dp[j] = min(dp[j], dp[j - i] + 1)
        return dp[amount]
    #End here
```

	Test	Input	Expected	Got	
✔	ob1.coinChange(s,amt)	3 11 1 2 5	3	3	✔
✔	ob1.coinChange(s,amt)	3 12 1 2 5	3	3	✔
✔	ob1.coinChange(s,amt)	3 22 1 2 5	5	5	✔

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 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):
    ############   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 = []
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	✔

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.

Question **3**

Correct

Mark 20.00 out of 20.00

Write a python program to find the maximum contiguous subarray.

For example:

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

Answer: (penalty regime: 0 %)

Reset answer

```
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()))
print("Maximum contiguous sum is", maxSubArraySum(a,n))
```

	Test	Input	Expected	Got	
✔	maxSubArraySum(a,n)	8 -2 -3 4 -1 -2 1 5 -3	Maximum contiguous sum is 7	Maximum contiguous sum is 7	✔
✔	maxSubArraySum(a,n)	5 1 -2 -3 4 5	Maximum contiguous sum is 9	Maximum contiguous sum is 9	✔

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 Implement Minimum cost path using Dynamic Programming.

For example:

Input	Result
3 3	8

Answer: (penalty regime: 0 %)

```
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 **5**

Not answered

Mark 0.00 out of 20.00

Write a python program for the implementation of merge sort on the given list of values.

For example:

Input	Result
5 12 10 61 2 3	Given array is 12 10 61 2 3 Sorted array is 2 3 10 12 61
6 20 10 31 49 87 6	Given array is 20 10 31 49 87 6 Sorted array is 6 10 20 31 49 87

Answer: (penalty regime: 0 %)

```
1
```