Started on	Monday, 19 May 2025, 2:53 PM
State	Finished
Completed on	Monday, 19 May 2025, 5:21 PM
Time taken	2 hours 28 mins
Overdue	28 mins 5 secs
Grade	**80.00** out of 100.00

Question 1

Correct

Mark 20.00 out of 20.00

Create a python program to find the minimum number of jumps needed to reach end of the array using Dynamic Programming.

For example:

Test	Input	Result
minJumps(arr,n)	6 1 3 6 1 0 9	Minimum number of jumps to reach end is 3

Answer: (penalty regime: 0 %)

Reset answer

```
def minJumps(arr, n):
    jumps = [0 for i in range(n)]
    if (n == 0) or (arr[0] == 0):
        return float('inf')
    jumps[0] = 0
    for i in range(1, n):
        jumps[i] = float('inf')
        for j in range(i):
            if (i <= j + arr[j]) and (jumps[j] != float('inf')):
                jumps[i] = min(jumps[i], jumps[j] + 1)
                break
    return jumps[n-1]
arr = []
n = int(input())
for i in range(n):
    arr.append(int(input()))
print('Minimum number of jumps to reach','end is', minJumps(arr,n))

```

	Test	Input	Expected	Got	
✔	minJumps(arr,n)	6 1 3 6 1 0 9	Minimum number of jumps to reach end is 3	Minimum number of jumps to reach end is 3	✔
✔	minJumps(arr,n)	7 2 3 -8 9 5 6 4	Minimum number of jumps to reach end is 3	Minimum number of jumps to reach end is 3	✔

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.

Question **2**

Incorrect

Mark 0.00 out of 20.00

Create a python program to find the longest palindromic substring using optimal algorithm Expand around center.

For example:

Test	Input	Result
findLongestPalindromicSubstring(s)	samsunggnusgnusam	sunggnus

Answer: (penalty regime: 0 %)

Reset answer

```
def expand(s, low, high):
    length = len(s)

    while low >= 0 and high < length and s[low] == s[high]:
        low = low - 1
        high = high + 1

    return s[low + 1:high]


def findLongestPalindromicSubstring(s):

    if not s or not len(s):
        return ''

    ########### Add your code here ###########

if __name__ == '__main__':

    s = input()       #'mojologiccigolmojo'

    print(findLongestPalindromicSubstring(s))
```

	Test	Input	Expected	Got	
✖	findLongestPalindromicSubstring(s)	samsunggnusgnusam	sunggnus	None	✖

Some hidden test cases failed, too.

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

Show differences

Incorrect

Marks for this submission: 0.00/20.00.

Question **3**

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):
        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]
ob1 = Solution()
n=int(input())
s=[]
```

	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 **4**

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):
    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
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,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 **5**

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

```
import sys
V = 5
INF = sys.maxsize
def minimumCostSimplePath(u, destination,
                          visited, graph):
    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
    return ans
if __name__=="__main__":
    graph = [[INF for j in range(V)]
             for i in range(V)]
    visited = [0 for i in range(V)]
    graph[0][1] = -1
```

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

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.