Started on	Monday, 5 May 2025, 3:31 PM
State	Finished
Completed on	Monday, 5 May 2025, 3:41 PM
Time taken	39 mins 46 secs
Grade	**100.00** 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**

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 #############
    max_till_now = a[0]
    max_ending = 0

    for i in range(0, size):
        max_ending = max_ending + a[i]
        if max_ending < 0:
            max_ending = 0


        elif (max_till_now < max_ending):
            max_till_now = max_ending

    return max_till_now
n=int(input())
a =[]
for i in range(n):
    a.append(int(input()))

```

	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	✔

	Test	Input	Expected	Got	
✔	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 **3**

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):
    ########  Add your code 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
    return ans

if __name__=="__main__":
    graph = [[INF for j in range(V)]
                  for i in range(V)]
```

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

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.

Question **4**

Correct

Mark 20.00 out of 20.00

LONGEST PALINDROMIC SUBSEQUENCE

Given a sequence, find the length of the longest palindromic subsequence in it.

For example:

Input	Result
ABBDCACB	The length of the LPS is 5

Answer: (penalty regime: 0 %)

```
def Lps(X):
    n=len(X)
    dp=[[0 for _ in range(n)] for _ in range(n)]
    for x in range(n):
        dp[x][x]=1
    for l in range(2,n+1):
        for i in range(n-l+1):
            j=i+l-1
            if X[i]==X[j]:
                dp[i][j]=dp[i+1][j-1]+2
            else:
                dp[i][j]=max(dp[i+1][j],dp[i][j-1])
    return dp[0][n-1]
X=input()
print("The length of the LPS is",Lps(X))
```

	Input	Expected	Got	
✔	ABBDCACB	The length of the LPS is 5	The length of the LPS is 5	✔
✔	BBABCBCAB	The length of the LPS is 7	The length of the LPS is 7	✔
✔	cbbd	The length of the LPS is 2	The length of the LPS is 2	✔
✔	abbab	The length of the LPS is 4	The length of the LPS is 4	✔

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.

Question **5**

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 ############
       dp = [float('inf')] * (amount + 1)
       dp[0]=0
       for coin in coins:
           for i in range(coin, amount + 1):
               dp[i] = min(dp[i], dp[i - coin] + 1)
       return dp[amount] if dp[amount]!=float('inf') else -1

ob1 = Solution()
n=int(input())
s=[]
amt=int(input())
for i in range(n):
    s.append(int(input()))


print(ob1.coinChange(s,amt))
```

	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.