Started on	Wednesday, 23 July 2025, 10:14 AM
State	Finished
Completed on	Thursday, 7 August 2025, 10:30 AM
Time taken	15 days
Overdue	14 days 22 hours
Grade	**100.00** 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 ###########
        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! ✔

Marks for this submission: 20.00/20.00.

Question **2**

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)]
    visited = [0 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 **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)

For example:

Test	Input	Result
minJumps(arr, 0, n-1)	10 1 3 6 3 2 3 6 8 9 5	Minimum number of jumps to reach end is 4

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

	Test	Input	Expected	Got	
✔	minJumps(arr, 0, n-1)	10 1 3 6 3 2 3 6 8 9 5	Minimum number of jumps to reach end is 4	Minimum number of jumps to reach end is 4	✔
✔	minJumps(arr, 0, n-1)	7 3 2 5 9 4 1 6	Minimum number of jumps to reach end is 2	Minimum number of jumps to reach end is 2	✔

Passed all tests! ✔

Marks for this submission: 20.00/20.00.

Question **4**

Correct

Mark 20.00 out of 20.00

Given a string s, return *the longest palindromic substring* in s.

Example 1:

```
Input: s = "babad"
Output: "bab"
Explanation: "aba" is also a valid answer.
```

Example 2:

```
Input: s = "cbbd"
Output: "bb"
```

For example:

Test	Input	Result
ob1.longestPalindrome(str1)	ABCBCB	BCBCB

Answer: (penalty regime: 0 %)

Reset answer

```
class ob1:
    @staticmethod
    def longestPalindrome(s):
        start = 0
        max_len = 0
        result = ""
        for i in range(len(s)):
            l = r = i
            while l >= 0 and r < len(s) and s[l] == s[r]:
                if (r - l + 1) > max_len or ((r - l + 1) == max_len and s[l:r+1] < result):
                    start = l
                    max_len = r - l + 1
                    result = s[l:r+1]
                l -= 1
                r += 1
            l = i
            r = i + 1
            while l >= 0 and r < len(s) and s[l] == s[r]:
                if (r - l + 1) > max_len or ((r - l + 1) == max_len and s[l:r+1] < result):
                    start = l
                    max_len = r - l + 1
                    result = s[l:r+1]
```

	Test	Input	Expected	Got	
✔	ob1.longestPalindrome(str1)	ABCBCB	BCBCB	BCBCB	✔
✔	ob1.longestPalindrome(str1)	BABAD	ABA	ABA	✔

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 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(a,size):
        ####################  Add your Code here #############
        max_sum = A[0]
        current_sum = A[0]
        for i in range(1, len(A)):
            current_sum = max(A[i], current_sum + A[i])
            max_sum = max(max_sum, current_sum)
        return max_sum

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.