TT DS PYTHON MODULE-23

Started on	Saturday, 17 May 2025, 9:02 AM
State	Finished
Completed on	Saturday, 17 May 2025, 9:29 AM
Time taken	27 mins 5 secs
Grade	**80.00** out of 100.00

Question **1**

Correct

Mark 20.00 out of 20.00

Flag question

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):
        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.

Question **2**

Not answered

Mark 0.00 out of 20.00

Flag question

Create a python program to find the longest palindromic substring using Brute force method in a given string.

For example:

Input	Result
mojologiccigolmojo	logiccigol

Answer: (penalty regime: 0 %)

Reset answer

```python
def printSubStr(str, low, high):

    for i in range(low, high + 1):
        print(str[i], end = "")

def longestPalindrome(str):
    ############  Add your code here ###########

    printSubStr(str, start, start + maxLength - 1)

if __name__ == '__main__':

    str = input()

    longestPalindrome(str)
```

Question **3**

Correct

Mark 20.00 out of 20.00

Flag question

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

```python
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 **4**

Correct

Mark 20.00 out of 20.00

Flag question

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	The sum of contiguous sublist with the largest sum is 23.8

	-9.6 -3.5 6.3 8.31 9.2

Answer: (penalty regime: 0 %)

Reset answer

```
class Solution:
    def maxSubArray(self,A):
        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
	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

Passed all tests!

Correct

Marks for this submission: 20.00/20.00.

Question **5**

Correct

Mark 20.00 out of 20.00

Flag question

Create a python program to find Minimum number of jumps to reach end of the array using naive method(recursion) using float value

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

    ############   Add your code 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.

Finish r