| | |
|---|---|
| **Started on** | Monday, 19 May 2025, 10:24 AM |
| **State** | Finished |
| **Completed on** | Monday, 19 May 2025, 10:51 AM |
| **Time taken** | 27 mins 29 secs |
| **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<br>11<br>1<br>2<br>5 | 3 |

**Answer:** (penalty regime: 0 %)

Reset answer

```
class Solution(object):
    def coinChange(self, coins, amount):
      ####################      Add your Code 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]


```

| | Test | Input | Expected | Got | |
|---|---|---|---|---|---|
| ✔ | ob1.coinChange(s,amt) | 3<br>11<br>1<br>2<br>5 | 3 | 3 | ✔ |
| ✔ | ob1.coinChange(s,amt) | 3<br>12<br>1<br>2<br>5 | 3 | 3 | ✔ |
| ✔ | ob1.coinChange(s,amt) | 3<br>22<br>1<br>2<br>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 the minimum number of jumps needed to reach end of the array using Dynamic Programming.

**For example:**

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

**Answer:** (penalty regime: 0 %)

Reset answer

```
def minJumps(arr, n):
    ##########  Add your code here ##############
    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<br>1<br>3<br>6<br>1<br>0<br>9 | Minimum number of jumps to reach end is 3 | Minimum number of jumps to reach end is 3 | ✔ |
| ✔ | minJumps(arr,n) | 7<br>2<br>3<br>-8<br>9<br>5<br>6<br>4 | Minimum number of jumps to reach end is 3 | Minimum number of jumps to reach end is 3 | ✔ |

Passed all tests! ✔

Marks for this submission: 20.00/20.00.

Question **3**

Correct

Mark 20.00 out of 20.00

Given an integer array `nums`, find the contiguous subarray (containing at least one number) which has the largest sum and return *its sum*.

A **subarray** is a **contiguous** part of an array.

**Example 1:**

```
Input: nums = [-2,1,-3,4,-1,2,1,-5,4]
Output: 6
Explanation: [4,-1,2,1] has the largest sum = 6.
```

**For example:**

| Test | Input | Result |
|---|---|---|
| s.maxSubArray(A) | 9<br>-2<br>1<br>-3<br>4<br>-1<br>2<br>1<br>-5<br>4 | The sum of contiguous sublist with the largest sum is 6 |

**Answer:** (penalty regime: 0 %)

Reset answer

```
class Solution:
    def maxSubArray(self,A):
        ############ Add your Code here
        res=0
        mm= -10000
        for v in A:
            res+=v
            mm=max(mm,res)
            if res<0:
                res=0
        return mm


A =[]
n=int(input())
for i in range(n):
    A.append(int(input()))
s=Solution()
print("The sum of contiguous sublist with the largest sum is",s.maxSubArray(A))

```

| | Test | Input | Expected | Got | |
|---|---|---|---|---|---|
| ✔ | s.maxSubArray(A) | 9<br>-2<br>1<br>-3<br>4<br>-1<br>2<br>1<br>-5<br>4 | The sum of contiguous sublist with the largest sum is 6 | The sum of contiguous sublist with the largest sum is 6 | ✔ |

| | Test | Input | Expected | Got | |
|---|---|---|---|---|---|
| ✔ | s.maxSubArray(A) | 5<br>5<br>4<br>-1<br>7<br>8 | The sum of contiguous sublist with the largest sum is 23 | The sum of contiguous sublist with the largest sum is 23 | ✔ |

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.

Question **4**

Incorrect

Mark 0.00 out of 20.00

Write a python program to implement quick sort using the middle element as pivot on the list of given integer values.

**For example:**

| Input | Result |
|---|---|
| 8<br>6<br>3<br>5<br>1<br>2<br>9<br>8<br>7 | [1, 2, 3, 5, 6, 7, 8, 9] |

**Answer:** (penalty regime: 0 %)

```
def quicksort(arr):
    n=len(arr)
    mid=len(arr)//2
    pivot=arr[n//2]
    print(pivot)
    left=arr[:mid]
    right=arr[mid:]

    for i in range(len(left)):
        for j in range(len(right),0):
            for x in left:
                for y in right:
                    if x>pivot:
                        i+=1
                    if y<pivot:
                        j-=1
                    if i==j:
                        pivot=x



n=int(input())
```

| | Input | Expected | Got | |
|---|---|---|---|---|
| ✖ | 8<br>6<br>3<br>5<br>1<br>2<br>9<br>8<br>7 | [1, 2, 3, 5, 6, 7, 8, 9] | 2 | ✖ |

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 **5**

Correct

Mark 20.00 out of 20.00

Write a Python program using A Naive recursive implementation of Minimum Cost Path Problem.

**For example:**

| Input | Result |
|---|---|
| 3<br>3 | 8 |

**Answer:** (penalty regime: 0 %)

Reset answer

```
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<br>3 | 8 | 8 | ✔ |

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.