| | |
|---|---|
| **Started on** | Saturday, 17 May 2025, 9:24 AM |
| **State** | Finished |
| **Completed on** | Tuesday, 20 May 2025, 2:29 PM |
| **Time taken** | 3 days 5 hours |
| **Overdue** | 3 days 3 hours |
| **Grade** | **80.00** out of 100.00 |

Question **1**
Correct
Mark 20.00 out of 20.00

Write a Python program to sort unsorted numbers using Multi-key quicksort

**For example:**

| Test | Input | Result |
|---|---|---|
| `quick_sort_3partition(nums, 0, len(nums)-1)` | 5<br>4<br>3<br>5<br>1<br>2 | Original list:<br>[4, 3, 5, 1, 2]<br>After applying Random Pivot Quick Sort the said list becomes:<br>[1, 2, 3, 4, 5] |
| `quick_sort_3partition(nums, 0, len(nums)-1)` | 6<br>21<br>10<br>3<br>65<br>4<br>8 | Original list:<br>[21, 10, 3, 65, 4, 8]<br>After applying Random Pivot Quick Sort the said list becomes:<br>[3, 4, 8, 10, 21, 65] |

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

```python
                sorting[b], sorting[i] = sorting[i], sorting[b]
                b -= 1
            else:
                i += 1
        quick_sort_3partition(sorting, left, a - 1)
        quick_sort_3partition(sorting, b + 1, right)
def three_way_radix_quicksort(sorting: list) -> list:
    if len(sorting) <= 1:
        return sorting
    return (
        three_way_radix_quicksort([i for i in sorting if i < sorting[0]])
        + [i for i in sorting if i == sorting[0]]
        + three_way_radix_quicksort([i for i in sorting if i > sorting[0]])
    )
n = int(input())
nums = []
for _ in range(n):
    element = int(input())
    nums.append(element)
print("Original list:")
print(nums)
print("After applying Random Pivot Quick Sort the said list becomes:")
```

| | Test | Input | Expected | Got | |
|---|---|---|---|---|---|
| ✓ | `quick_sort_3partition(nums, 0, len(nums)-1)` | 5<br>4<br>3<br>5<br>1<br>2 | Original list:<br>[4, 3, 5, 1, 2]<br>After applying Random Pivot Quick Sort the said list becomes:<br>[1, 2, 3, 4, 5] | Original list:<br>[4, 3, 5, 1, 2]<br>After applying Random Pivot Quick Sort the said list becomes:<br>[1, 2, 3, 4, 5] | ✓ |
| ✓ | `quick_sort_3partition(nums, 0, len(nums)-1)` | 6<br>21<br>10<br>3<br>65<br>4<br>8 | Original list:<br>[21, 10, 3, 65, 4, 8]<br>After applying Random Pivot Quick Sort the said list becomes:<br>[3, 4, 8, 10, 21, 65] | Original list:<br>[21, 10, 3, 65, 4, 8]<br>After applying Random Pivot Quick Sort the said list becomes:<br>[3, 4, 8, 10, 21, 65] | ✓ |
| ✓ | `quick_sort_3partition(nums, 0, len(nums)-1)` | 4<br>21<br>3<br>10<br>4 | Original list:<br>[21, 3, 10, 4]<br>After applying Random Pivot Quick Sort the said list becomes:<br>[3, 4, 10, 21] | Original list:<br>[21, 3, 10, 4]<br>After applying Random Pivot Quick Sort the said list becomes:<br>[3, 4, 10, 21] | ✓ |

Passed all tests! ✓

Correct
Marks for this submission: 20.00/20.00.

Question **2**

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

```python
class Solution:
    def maxSubArray(self,A):
        ############ 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(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 | ✓ |
| ✓ | 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 **3**

Correct

Mark 20.00 out of 20.00

## Print All Paths With Minimum Jumps

```
1. You are given a number N representing number of elements.
2. You are given N space separated numbers (ELE : elements).
3. Your task is to find & print
    3.1) "MINIMUM JUMPS" need from 0th step to (n-1)th step.
    3.2) all configurations of "MINIMUM JUMPS".
NOTE: Checkout sample question/solution video inorder to have more insight.
```

**For example:**

| Test | Input | Result |
|---|---|---|
| `minJumps(arr)` | 10<br>3<br>3<br>0<br>2<br>1<br>2<br>4<br>2<br>0<br>0 | 0 -> 3 -> 5 -> 6 -> 9<br>0 -> 3 -> 5 -> 7 -> 9 |

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

Reset answer

```python
from queue import Queue
import sys
class Pair(object):
    idx = 0
    psf = ""
    jmps = 0
    def __init__(self, idx, psf, jmps):

        self.idx = idx
        self.psf = psf
        self.jmps = jmps
def minJumps(arr):
    MAX_VALUE = sys.maxsize
    dp = [MAX_VALUE for i in range(len(arr))]
    n = len(dp)
    dp[n - 1] = 0

    for i in range(n - 2, -1, -1):
        steps = arr[i]
        minimum = MAX_VALUE

        for j in range(1, steps + 1, 1):
```

| | Test | Input | Expected | Got | |
|---|---|---|---|---|---|
| ✓ | `minJumps(arr)` | 10<br>3<br>3<br>0<br>2<br>1<br>2<br>4<br>2<br>0<br>0 | 0 -> 3 -> 5 -> 6 -> 9<br>0 -> 3 -> 5 -> 7 -> 9 | 0 -> 3 -> 5 -> 6 -> 9<br>0 -> 3 -> 5 -> 7 -> 9 | ✓ |
| ✓ | `minJumps(arr)` | 7<br>5<br>5<br>0<br>3<br>2<br>3<br>6 | 0 -> 1 -> 6<br>0 -> 3 -> 6<br>0 -> 4 -> 6<br>0 -> 5 -> 6 | 0 -> 1 -> 6<br>0 -> 3 -> 6<br>0 -> 4 -> 6<br>0 -> 5 -> 6 | ✓ |

Passed all tests! ✓

Correct

Marks for this submission: 20.00/20.00.

Question **4**

Correct

Mark 20.00 out of 20.00

Create a Dynamic Programming  python Implementation  of Coin Change Problem.

**For example:**

| Test | Input | Result |
|---|---|---|
| `count(arr, m, n)` | 3<br>4<br>1<br>2<br>3 | 4 |

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

Reset answer

```
def count(S, m, n):
    table = [[0 for x in range(m)] for x in range(n+1)]
    for i in range(m):
        table[0][i] = 1
    for i in range(1, n+1):
        for j in range(m):

            x = table[i - S[j]][j] if i-S[j] >= 0 else 0

            y = table[i][j-1] if j >= 1 else 0

            table[i][j] = x + y

    return table[n][m-1]


arr = []
m = int(input())
n = int(input())
for i in range(m):
    arr.append(int(input()))
print(count(arr, m, n))
```

| | Test | Input | Expected | Got | |
|---|---|---|---|---|---|
| ✓ | `count(arr, m, n)` | 3<br>4<br>1<br>2<br>3 | 4 | 4 | ✓ |
| ✓ | `count(arr, m, n)` | 3<br>16<br>1<br>2<br>5 | 20 | 20 | ✓ |

Passed all tests! ✓

Correct

Marks for this submission: 20.00/20.00.

Question **5**

Not answered

Mark 0.00 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

```python
import sys
V = 5
INF = sys.maxsize
def minimumCostSimplePath(u, destination,
                          visited, graph):
########  Add your code here ##############

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
    graph[0][3] = 1
    graph[1][2] = -2
    graph[2][0] = -3
    graph[3][2] = -1
    graph[4][3] = 2
    s = 0
    t = 2
    visited[s] = 1
    print(minimumCostSimplePath(s, t, visited, graph))

```