**Java - Competitive Programming Challenge**

**Question 7**

There is a shop with old-style cash registers. Rather than scanning items and pulling the price from a database, the price of each item is typed in manually. This method sometimes leads to errors. Given a list of items and their correct prices, compare the prices to those entered when each item was sold. Determine the number of errors in selling prices.

**Example:**

products = ['eggs', 'milk', 'cheese’]

productPrices = [2.89, 3.29, 5.79]

productSold = ['eggs', 'eggs', 'cheese', 'milk']

soldPrice = [2.89, 2.99, 5.97, 3.29].

Price

Product Actual Expected Error

eggs 2.89 2.89

eggs 2.99 2.89 1

cheese 5.97 5.79 1

milk 3.29 3.29

The second sale of eggs has a wrong price, as does the sale of cheese. There are 2 errors in pricing.

**Function Description:**

Complete the function priceCheck in the editor below.

priceCheck has the following parameter(s):

* string products[n]: each products[i] is the name of an item for sale string productPrices[n]: each productPrices[i] is the price of products[i]
* string productSold[m]: each productSold[j] is the name of a product sold
* float soldPrice[m]: each soldPrice[j] contains the sale price recorded for productSold[j].

**Returns:**

* int: the number of sale prices that were entered incorrectly

**Constraints:**

* 1 ≤ n ≤ 105
* 1 ≤ m ≤ n

1. ≤ productPrices[i], soldPrice[j] ≤ 100000.00, where 0 ≤ i < n, and 0 ≤ j < m

**Question 8**

**Question Description:**

Ryan is movie obsessed and has collected a list of movie quality ratings. He wants to watch the largest contiguous list of movies with the highest cumulative ratings possible. To do this, he must calculate the sum of all contiguous subarrays in order to determine the maximum possible subarray sum.

For example, ratings are arr = [-1,3,4,-2,5,-7]. We can see that the highest value contiguous subarray runs from arr[1]-arr[4] and is 3 + 4 + -2 + 5 = 10.

**Function Description:**

Complete the function maximumSum in the editor below. It must return a long integer denoting the maximum sum for any contiguous subarray in arr.

maximumSum has the following parameter(s):

* arr[arr[0],...arr[n-1]]: an array of integers

**Constraints:**

* 1 ≤ n ≤ 106

−107 ≤ arr[i] ≤ 107

**Question 9**

**Question Description:**

Given an interface named "OnlineAccount" that models the account of popular online video streaming platforms, perform the operations listed below. The interface "OnlineAccount" consists of the basePrice, regularMoviePrice, and exclusiveMoviePrice.

In order to complete this challenge, you need to implement an incomplete class named "Account" which implements the "OnlineAccount" interface as well as the "Comparable<Account>" interface.

Class Account has two attributes to keep track of the number of movies watched:

1. Integer noOfRegularMovies
2. Integer noOfExclusiveMovies
3. String ownerName

Methods to complete for class Account:

1. Add a parameterized constructor that initializes the attributes ownerName, numberOfRegularMovies and numberOfExclusiveMovies.
2. double monthlyCost() => This method returns the monthly cost for the account. [Monthly Cost = base price + noOfRegularMovies\*regularMoviePrice + noOfExclusiveMovies\*exclusiveMoviePrice]
3. Override the compareTo method of the Comparable interface such that two accounts can be compared based on their monthly cost.
4. String toString() which returns => "Owner is [ownerName] and monthly cost is [monthlyCost] USD."