Q1 Balanced Array —- Coding

QUESTION DESCRIPTION

Given an array of numbers, find the index of the smallest array element (the pivot), for which the sums of all elements to the left and to the right are equal. The array may not be reordered.

Example arr=[1,2,3,4,6]

the sum of the first three elements, 1+2+3=6. The value of the last element is 6.
Using zero based indexing, arr[3]=4 is the pivot between the two subarrays.
The index of the pivot is 3.

Function Description

Complete the function balancedSum in the editor below.

balancedSum has the following parameter(s):

int arr[n]: an array of integers
Returns:
int: an integer representing the index of the pivot

Constraints
3 ≤ n ≤ 10
1 ≤ arr[i] ≤ 2 × 10 , where 0 ≤ i < n
It is guaranteed that a solution always exists.

Q2 Scoring System — DbRank

QUESTION DESCRIPTION

The math scores of each student have been stored in the STUDENT table. Write a query to print the ID and the NAME of each of the three highest scoring students. Print the NAMEs in descending order by SCORE, then ascending order by ID for matching SCOREs.

Input Format

STUDENT		
Name	Type	Description

ID	Integer	A student ID in the inclusive range [1, 1000]. This field is the primary key.
NAME	String	A student name. This field contains between 1 and 100 characters (inclusive).
SCORE	Flat	The Math score of the student.

Output Format

The result should contain the IDs and the NAMEs of the three highest scoring students. Print the records in descending order by SCORE, then ascending order by ID for matching SCOREs.

STUDENT.ID STUDENT.NAME

Sample Input

STUDENT		
ID	NAME	SCORE
1	Bob	50
2	John	65.5
3	Harry	45
4	Dick	85
5	Dev	25
6	Sid	98
7	Tom	90
8	Julia	70.5
9	Erica	81
10	Jerry	85

Sample Output
6 Sid
7 Tom
4 Dick

Explanation

The students are arranged in the descending order of their math scores, followed by the ascending order of their ids, as shown below:

Sid > Tom > Dick > Jerry > Erica > Julia > John > Bob > Harry > Dev

Dick's and Jerry's scores were the same, so they are shown in ID order.

Q3 Account Comparisons —- Coding

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" 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. Int 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."

Q4
Design and implement a producer consumer model using threads and basic data structures. (1-producer, 1-consumer model). Write the complete implementation in core java. (Plz avoid using java executor framework and advanced data structures).

Q5
Design and implement a stack using queues. Write the complete implementation in core java. (Plz avoid using java executor framework and advanced data structures).

Submission
Submit the link to your github repo on the following mail address.
- ananya.srivastava@baazigames.com