Assessments for Freshers drive held on 6 Apr 2019

You can choose any programming language to solve the given problem statements.

DO NOT WRITE CODE FOR GETTING INPUT FROM THE USER OR SHOWING OUTPUT TO THE USER. ONLY WRITE THE BODY OF THE FUNCTION ASKED

QUESTION-1

[Box](https://en.wikipedia.org/wiki/Magic_acid) Sorter

You have got colored boxes (cube shaped) of different size. Colors are from VIBGYOR (Violet, Indigo, Blue, Green, Yellow, Orange, Red). You need to write a function that can sort the given collection of boxes in the following order

1. If 2 boxes are of different sizes, bigger box should come before smaller box
2. If 2 boxes are of same size, then box should be returned in the order of VIBGYOR (Violet should come before Indigo, Indigo should be before Blue and so on)
3. If 2 boxes are of same size and same color, return in the order given to you

enum Color {

VIOLET, INDIGO, BLUE, GREEN, YELLOW, ORANGE, RED

}

public class CubeBox {

Color color;

Float width;

}

Implement the following function

public Box[] sort(Box[] inputBoxes) {

//write your code here

}

QUESTION-2

[IPL](https://en.wikipedia.org/wiki/Magic_acid) Fixture for League Matches

There are 8 Indian Premier League teams from 8 cities in India. We need to write a function that can generate a fixture table for League matches of these 8 teams. Following are the rules that need to be adhered while generating the Fixture table.

1. Each team has to play with every other team twice – once in their home city and another match in the other team’s city
2. A team has to get rest for AT LEAST 2 DAYS between the matches
3. To simplify the programming logic, let us assume that we will have only one match per day
4. Input to the function will be the first fixture that will specify start date, start team1 and start team 2. So that the first match can act as a trigger to decide the rest of the match
5. Function can order the matches in their own way and there is no specific ordering of teams to play the league matches
6. Match can be scheduled on any of the 7 days in a week

public class IplTeam {

String teamName;

String city;

}

public class Fixture {

Date matchDate;

IplTeam team1;

IplTeam team2;

}

Implement the following function:

public Fixture[] generateLeagueFixture(

IplTeam[] iplTeams, Fixture startingFixture) {

//write your code here

}

QUESTION-3

Largest Number of Orders

A company maintains information about its orders in table called *ORDERS*. Write a query to find the customer ID of the customer who has placed the largest number of orders. If there is more than one customer with the same number of orders, then print the one with the smallest customer ID.

**Input Format**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **ORDERS** | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| **Name** | **Type** | **Description** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| ID | Integer | A number in the inclusive range *[1, 1000]* which uniquely identifies the order. This is the primary key. |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| ORDER\_DATE | Date | The date when the order was placed. |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| STATUS | String | This is the order status. It can be PLACED, SHIPPED, IN TRANSIT, DELIVERED. |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| CUSTOMER\_ID | Integer | A number in the inclusive range *[1, 1000]* which uniquely identifies the customer who placed the order. |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

**Output Format**

The output of the query should be the customer ID from the *Orders* table for the customer who has placed the largest number of orders. If there is more than one customer with the same number of orders, then print the one with the smallest customer ID.

ORDERS.CUSTOMER\_ID

**Sample Input**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **ORDERS** | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| **ID** | **ORDER\_DATE** | **STATUS** | **CUSTOMER\_ID** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 1 | 2003-01-06 | PLACED | 363 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 2 | 2003-01-06 | PLACED | 128 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 3 | 2003-01-06 | IN TRANSIT | 121 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 4 | 2003-01-06 | DELIVERED | 121 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 5 | 2003-03-06 | PLACED | 128 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

**Sample Output**

121

**Explanation**

The customers with customer ID *121* and *128* have placed the largest number of orders i.e two orders. So, we choose the one with smallest customer ID i.e *121*.