1. Create a table **stock** using below data, The result should be

```
item year month qty rnk
---------- ----------
1001 2020  1     890 1
1001 2020  2     832 2
1001 2019  11    340 3
1002 2019  12    603 1
1002 2020  3     172 2
1002 2020  4     150 3
1002 2019  11    120 4
```

write a query to retrieve the date and the qty for the row(s) with the ***second highest*** qty in the entire table. Your query should work without needing any modification to give the second highest result for *any* set of data in the table, not just for the data given. Use the stock table in your query. The result should be

```
  dt          qty
---------- ----------
2019-12-16   402
```

2. Using the table called **journey** and the sample data given below, write a query to display the following result. You can assume that the **descr** column always contains a hyphen between the start and end point and that the **distance** column always ends with either “mi” or “km”, giving the distance in either miles or kilometers. In your answer, multiply the miles by 1.609 to convert the miles to km for the distance_km column.

```
CREATE TABLE journey (descr VARCHAR(100) PRIMARY KEY, distance VARCHAR(7) NOT
NULL);

INSERT INTO journey (descr, distance)
VALUES
 ('London-Paris','342km')
,('New York-Los Angeles','2445mi')
,('New York-Washington','203mi')
,('Paris-Rome',' 1106km');
```

Output:

```
journey_start      journey_end      distance_km
--------------   --------------     ------------
London              Paris              342.000
New York            Los Angeles        3934.005
New York            Washington         326.627
Paris               Rome               1106.000
```

3. (a)Write a class named Calculator. The class must contain methods add, subtract, multiply and divide with two input parameters. Create an object of Calculator class in another class named UseCalculator to perform the above mentioned operations. Numbers should be input through UseCalculator class.

(b). Write a class Area with method calculateArea. Method should return the area of the
rectangle and circle. Use polymorphism.

4. Create a class called Invoice that a hardware store might use to represent an invoice for an item sold at the store. An Invoice should include four pieces of information as instance variable

a. Part number(type String)
b. Part description(type String)
c. Quantity of the item being purchased (type int)
d. Price per item (double).

Your class should have a constructor that initializes the four instance variables. Provide a set and a get method for each instance variable. In addition, provide a method named **getInvoice** Amount that calculates the invoice amount (i.e., multiplies the quantity by the price per item), then returns the amount as a double value. If the quantity is not positive, it should be set to 0. If the price per item is not positive, it should be set to 0.0. Write a test application named InvoiceTest that demonstrates class Invoice's capabilities.

5. Write a program to count the total number of notes in a given amount.
Example:
Input amount: 575

Output:
Total number of notes:
500: 1
100: 0
50: 1
20: 1
10: 0
5: 1
2: 0
1: 0

6. find the peak elements in the given array, an element is called peak element if its value is not smaller than its adjacent values.

Input:
Array1 = [1,2,3,2,1,9,8]

Output:
3,9

7. Write a program to print all the **LEADERS** in an array. An element is leader if it is **greater** than all the elements to its right side. The rightmost element is always a leader. Create a new array with all the leaders from the input array.

Input:

[16, 17, 4, 3, 5, 2, 1, 2, 3, 4, 0]

Output:

[17, 5, 2, 4, 0]

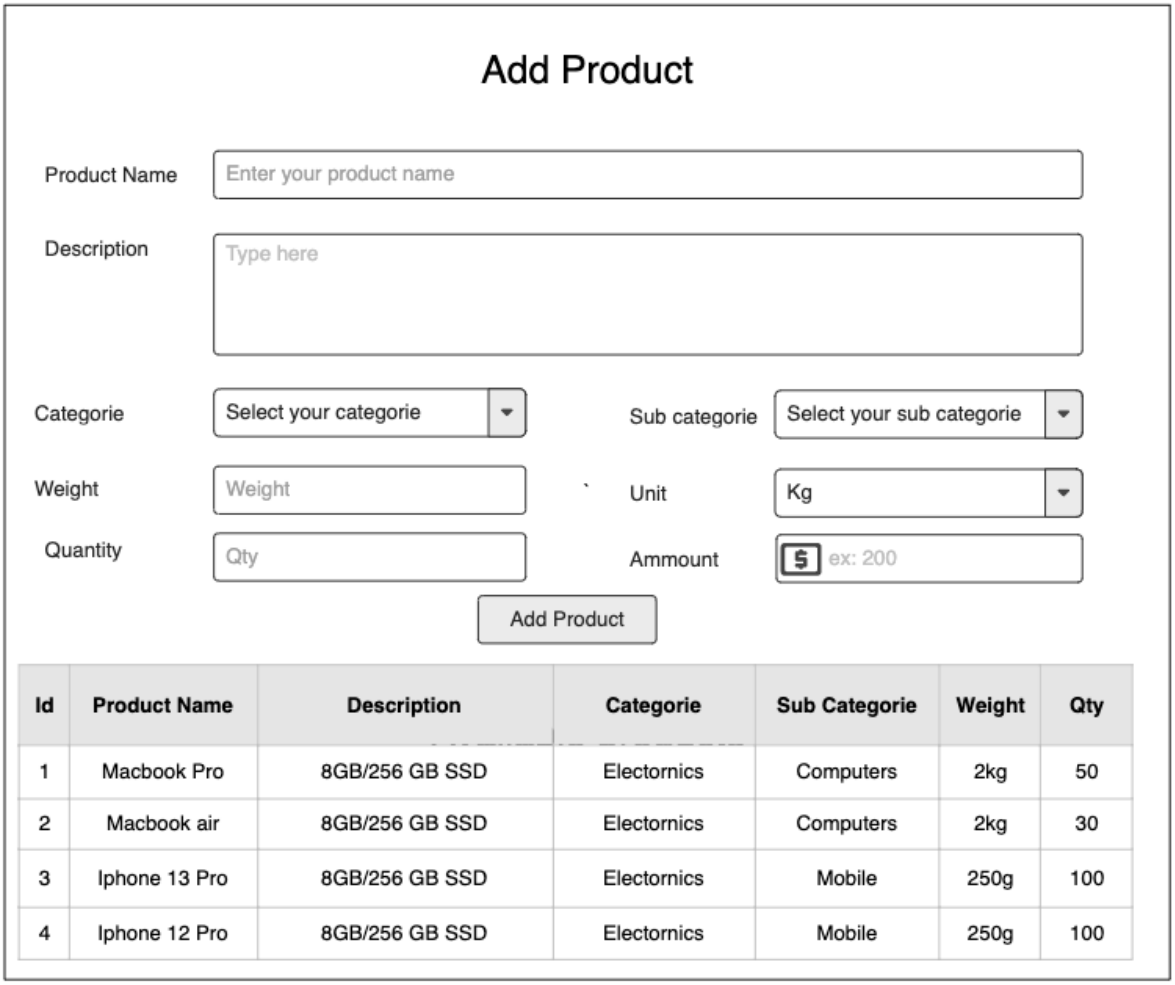
8. Given 3 arrays in increasing order. Find the elements that are common in all the 3 arrays.

Input;
Array1 = [10,11,12,13,14,15]
Array2 = [12,13,14,15,16]
Array3 = [14,15,16,17,18]

Output:
14,15

9. Create a product billing application for a Shop using HTML and Javascript as shown below. Users can add products and bill the products using the application. Let's design our application as shown like a wireframe. **You should use css for styling.**

Add product

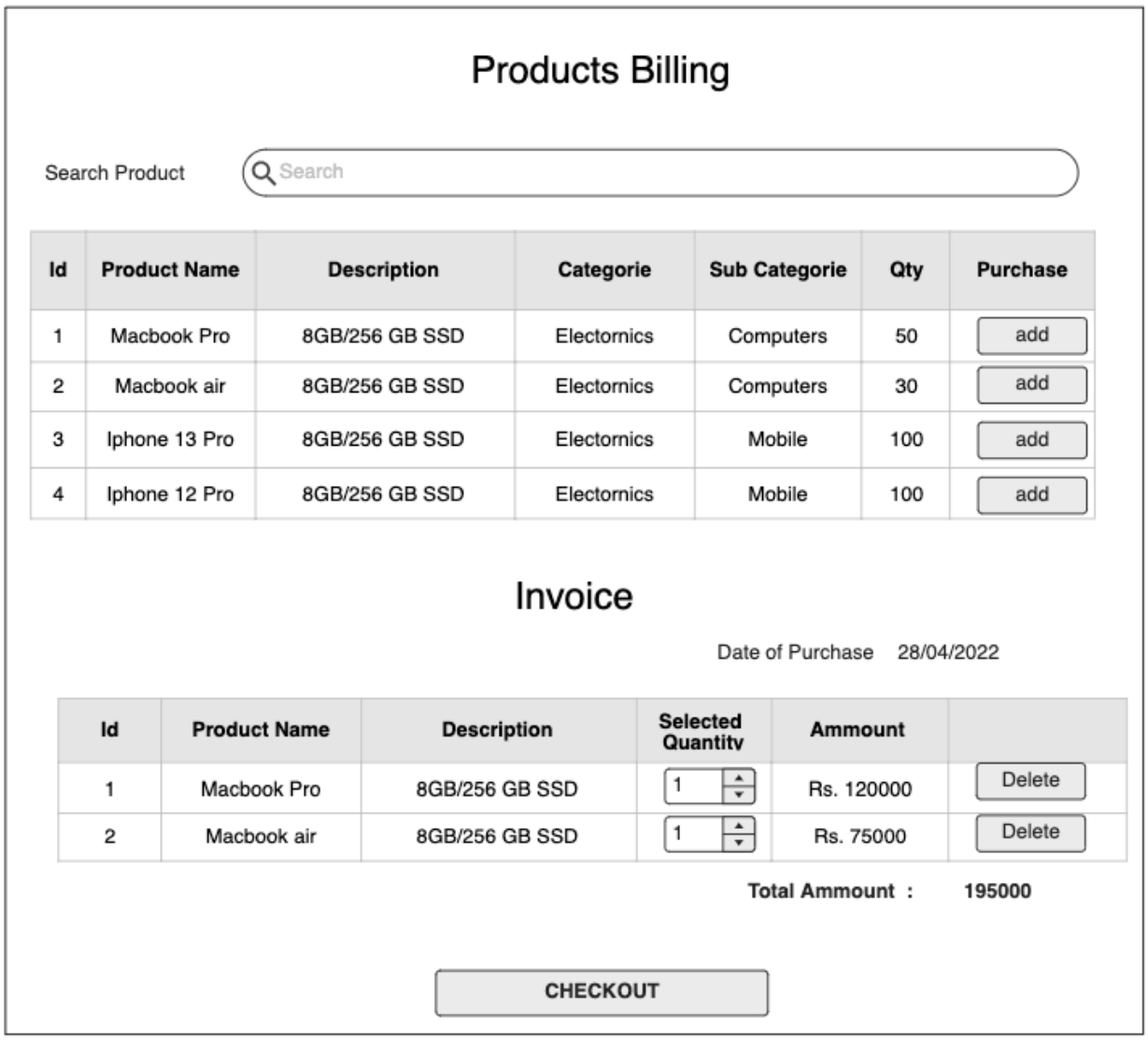


Id	Product Name	Description	Categorie	Sub Categorie	Weight	Qty
1	Macbook Pro	8GB/256 GB SSD	Electornics	Computers	2kg	50
2	Macbook air	8GB/256 GB SSD	Electornics	Computers	2kg	30
3	Iphone 13 Pro	8GB/256 GB SSD	Electornics	Mobile	250g	100
4	Iphone 12 Pro	8GB/256 GB SSD	Electornics	Mobile	250g	100

Tasks:

- Users can add product details using html forms.
- Give proper validations for adding product form.
- Display the product listing table below the add product form.
- After adding a product, Append that product details into the table given below.
- Also store the product details into local storage for future access.
- While loading the page, **fetch the product details from the local storage and display in the table.**

Product Billing



Tasks:

- Display the product details from the local storage into the 1st table.
- Users can search the products using the search box.
- Users can select the product from the table while purchasing.
- Create a table for invoice and append the selected product details into the table.
- Users can update the product quantity from the invoice table, also calculate the amount of products and total amount as shown above.
- While clicking on the checkout button, minimize the selected product quantities from the local storage and clear the invoice table.

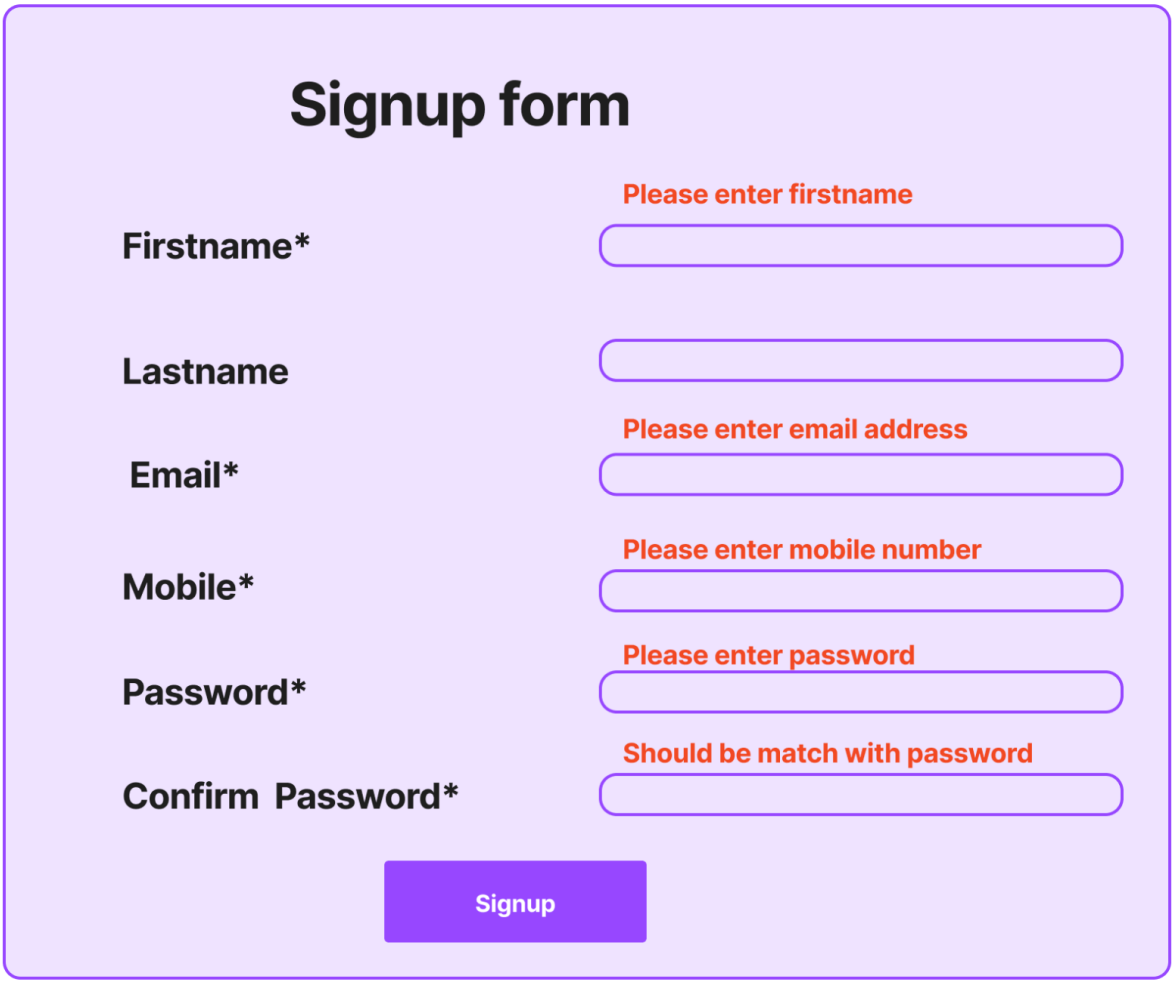
10. Design a sign-up form, a login form, and home page **(use css)**. When an account is created by the user by entering the details in the signup form correctly, it will be redirected to the login page.

When the user enters the username and password correctly on the login page, the user will be redirected to the home page.

When you click the login button on the login form, it will check whether the username and password you entered is correct or not. If you enter the correct login information, it will be redirected to the home page.If you have entered incorrect login details more than **three** times, the alert box will appear as shown below, and you cannot submit the form again.

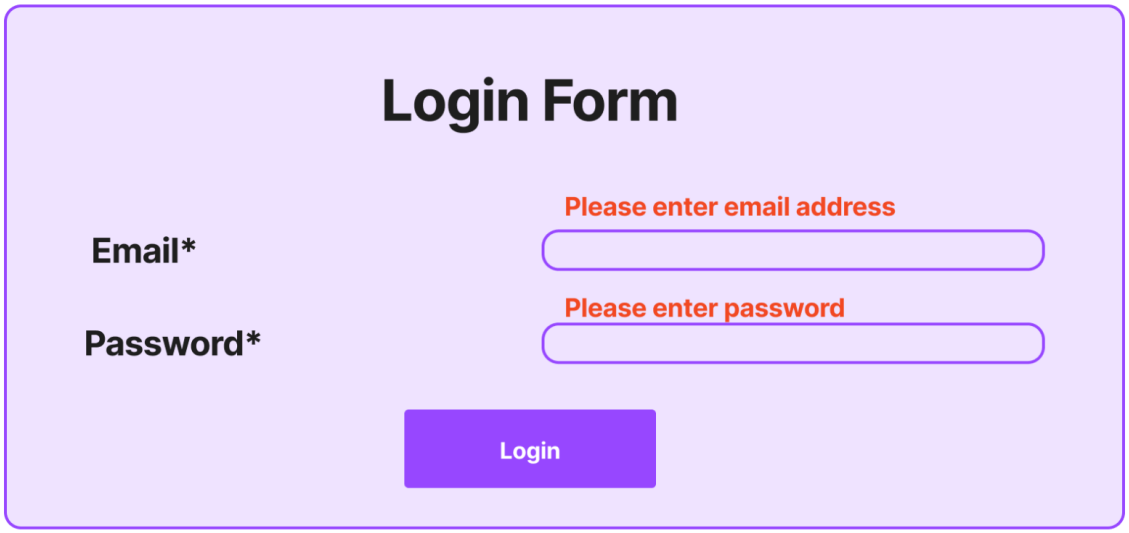
Signup form Tasks:

- Design a sign-up form.
- Signup form should contain first name, last name, mobile, email, password and confirm password.
- First name is mandatory, must contain at least 3 letters, special characters not allowed & last name is optional and doesn't allow special characters.
- Mobile numbers only accept 10 digits.
- Email address only accepts valid email address format.
- First letter of the password should be capitalized. Passwords must contain a special character (@, $, !, &, etc). Password length must be greater than 8 characters. One of the most important is that the password fields should not be empty.
- Passwords must match with the confirmed password.
- Give proper validation messages to each input
- Store registered user details into local storage.
- After registration completes, redirect to the login page.



Login form Tasks:

- Design a login form.
- Login form should contain username and password fields.
- Username should be the email address. Username only accepts valid email address format.
- Check username and password matching and redirect to the home page.


Login Form
Please enter email address
Email*
Please enter password
Password*
Login

Home Page:


THIS IS HOME PAGE