Q1: Electricity Bill Calculation

An electricity board charges the following rates to domestic users to discourage large consumption of energy:

- For the first 100 units: ₹0.60 per unit
- For the next 200 units: ₹0.80 per unit
- Beyond 300 units: ₹0.90 per unit
- All users are charged a **minimum of ₹50**
- If the total amount exceeds **₹300**, then an **additional surcharge of 15%** is added.

Implement a C++ program to read the names of users and number of units consumed and display the charges with names.

PROGRAM:

```
#include <iostream>
#include <string>
using namespace std;

int main() {
   string name;
   int units;
   float amount = 0, surcharge = 0, total;

   cout << "Enter customer name: ";
   getline(cin, name);

   cout << "Enter total units consumed: ";
   cin >> units;

   // Calculate charges based on units
   if (units <= 100) {
      amount = units * 0.60;
   } else if (units <= 300) {
      amount = 100 * 0.60 + (units - 100) * 0.80;
   } else {
      amount = 100 * 0.60 + 200 * 0.80 + (units - 300) * 0.90;
   }

   // Apply minimum charge
   if (amount < 50) {
      amount = 50;
   }

   // Add surcharge if amount > 300
   if (amount > 300) {
      surcharge = amount * 0.15;
   }

   total = amount + surcharge;
```

```
    // Output the bill
    cout << "\n--- Electricity Bill ---" << endl;
    cout << "Customer Name   : " << name << endl;
    cout << "Units Consumed  : " << units << endl;
    cout << "Total Bill (Rs) : " << total << endl;

    return 0;
}
```

SAMPLE OUTPUT:

```
Enter customer name: Pawan Joshi
Enter total units consumed: 460

--- Electricity Bill ---
Customer Name   : Pawan Joshi
Units Consumed  : 460
Total Bill (Rs)      : 418.6
```

Q2: Remove Character from String
Construct a C++ program that removes a specific character from a given string and returns the updated string.

Typical Input:
computer science is the future
Character to remove:
e
Expected Output:
computr scinc is th futur

PROGRAM:

```
#include <iostream>
#include <string>
using namespace std;

int main() {
   string str, result = "";
   char ch;

   cout << "Enter a string: ";
   getline(cin, str); // Read the entire line with spaces

   cout << "Enter character to remove: ";
   cin >> ch;

   // Remove all occurrences of the character
   for (int i = 0; i < str.length(); i++) {
      if (str[i] != ch) {
         result += str[i];
      }
   }

   cout << "Updated string: " << result << endl;

   return 0;
}
```

SAMPLE OUTPUT:

```
Enter a string: computer science is the future
Enter character to remove: e
Updated string: computr scinc is th futur
```

Q3: Write a C++ program that takes a string input from the user and replaces all occurrences of the word "dog" with "cat". Use the built-in string::find() and string::replace() functions to achieve this. Display the updated string as output.

```
#include <iostream>
#include <string>
using namespace std;

int main() {
    string text;
    cout << "Enter a string: ";
    getline(cin, text);   // take full line input

    size_t pos = 0;  // position in string
    while ((pos = text.find("dog", pos)) != string::npos) {
        text.replace(pos, 3, "cat");  // replace "dog" (3 letters) with "cat"
        pos += 3; // move forward to avoid infinite loop
    }

    cout << "Updated string: " << text << endl;
    return 0;
}
```

SAMPLE OUTPUT:

Enter a string: The dog is barking. That dog is loud. I like dogs.
Updated string: The cat is barking. That cat is loud. I like dogs.

Q4. Write a C++ program to check whether the two numbers entered by the user are co-prime or not.
Two numbers are said to be co-prime if they have no common factor other than 1.
Use a for loop to check for common divisors and print "Co-Prime" if the numbers are co-prime, otherwise print "Not Co-Prime".

```
#include <iostream>
using namespace std;

int main() {
   int a, b;
   cout << "Enter two numbers:\n";
   cin >> a >> b;

   bool isCoPrime = true;

   for (int i = 2; i <= (a < b ? a : b); i++) {
      if (a % i == 0 && b % i == 0) {
         isCoPrime = false;
         break;
      }
   }

   if (isCoPrime)
      cout << "Co-Prime\n";
   else
      cout << "Not Co-Prime\n";

   return 0;
}
```

SAMPLE OUTPUT:

```
Enter two numbers:
8 15
Co-Prime

Enter two numbers:
12 18
Not Co-Prime
```

Q5. Write a C++ program to display "Hello World" on the screen.
The program should print the message "Hello World" as output.

```
#include <iostream>
using namespace std;

int main() {
   cout << "Hello World\n";
   return 0;
}
```

SAMPLE OUTPUT:

```
Hello World
```

Q6. Write a C++ program to take a line of text as input using getline() and display it back to the user.
The program should read a full line of input (including spaces) and then print it.

```
#include <iostream>
#include <string>
using namespace std;

int main() {
   string text;
   cout << "Enter a line of text:\n";
   getline(cin, text);

   cout << "You entered: " << text << "\n";
   return 0;
}
```

SAMPLE OUTPUT:

```
Enter a line of text:
I am learning C++
You entered: I am learning C++
```

Q7. You are given an array of elements. Now you need to choose the best index of this array. An index of the array is called best if the special sum of this index is maximum across the special sums of all the other indices.
To calculate the special sum for any index:

- **Pick the first element at that index and add it to your sum.**
- **Then pick the next two elements and add them to your sum.**
- **Then pick the next three elements and add them to your sum.**
- **Continue this process, increasing the group size by 1 each time, until it is no longer possible to pick the required number of elements.**

Find the best index and, in the output, print its corresponding special sum.
Note:

- **There may be more than one best index, but you only need to print the maximum special sum.**

Input:

- **First line contains an integer — the number of elements in the array.**
- **Next line contains space-separated integers denoting the elements of the array.**

Output:

- **Print an integer that denotes the maximum special sum.**

Input/Output Format	
Typical Input	**Expected Output**
5 1 3 1 2 5	8
10 2 1 3 9 2 4 -10 -9 1 3	9

```
#include <iostream>
using namespace std;

int main() {
   int n;
   cin >> n;

   int arr[1000];
   for (int i = 0; i < n; i++) {
      cin >> arr[i];
   }

   int maxSum = -1000000;
```

```
    for (int i = 0; i < n; i++) {
        int sum = 0;
        int groupSize = 1;
        int index = i;

        while (index + groupSize - 1 < n) {
            for (int j = 0; j < groupSize; j++) {
                sum += arr[index + j];
            }
            index += groupSize;
            groupSize++;
        }

        if (sum > maxSum) {
            maxSum = sum;
        }
    }

    cout << maxSum << endl;
    return 0;
}
```

SAMPLE OUTPUT:

```
5
1 2 3 4 5

11
```

Q8. A cinema hall wants to develop a small program in C++ seating arrangement for cinema. The hall contains 5 rows (10 seats each) = 50 seats available.
The cinema hall management must have full functionality:

1. **Initialization of seats**
2. **Booking a seat**
3. **Cancel booking**
4. **Display the seating arrangement**

You need to design a structure for cinema hall. It contains (2D array) for seating arrangement.
Additional data to be maintained:

- **Hall name**
- **Total seats**
- **Booking seat count**

Note:

- **Pass the structures by reference to a function for operations like initialization, display, cancel, booking.**
- **Write a perfect C++ program using appropriate arrays and structures.**
- **Must be practical for a mini project.**

```
#include <iostream>
using namespace std;

struct CinemaHall {
   string hallName;
   int totalSeats;
   int bookedSeats;
   int seats[5][10];
};

void initializeSeats(CinemaHall &hall) {
   for (int i = 0; i < 5; i++) {
      for (int j = 0; j < 10; j++) {
         hall.seats[i][j] = 0;
      }
   }
   hall.bookedSeats = 0;
}

void bookSeat(CinemaHall &hall, int row, int col) {
   if (row < 0 || row >= 5 || col < 0 || col >= 10) {
      cout << "Invalid seat position!\n";
      return;
   }
   if (hall.seats[row][col] == 0) {
      hall.seats[row][col] = 1;
      hall.bookedSeats++;
      cout << "Seat booked successfully!\n";
   } else {
      cout << "Seat already booked!\n";
```

```
    }
}

void cancelSeat(CinemaHall &hall, int row, int col) {
    if (row < 0 || row >= 5 || col < 0 || col >= 10) {
        cout << "Invalid seat position!\n";
        return;
    }
    if (hall.seats[row][col] == 1) {
        hall.seats[row][col] = 0;
        hall.bookedSeats--;
        cout << "Booking cancelled!\n";
    } else {
        cout << "Seat was not booked!\n";
    }
}

void displaySeats(const CinemaHall &hall) {
    cout << "\nSeating Arrangement \n";
    for (int i = 0; i < 5; i++) {
        for (int j = 0; j < 10; j++) {
            cout << hall.seats[i][j] << " ";
        }
        cout << endl;
    }
    cout << "Total Seats: " << hall.totalSeats
         << " | Booked: " << hall.bookedSeats
         << " | Available: " << hall.totalSeats - hall.bookedSeats << endl;
}

int main() {
    CinemaHall hall;
    hall.hallName = "Global Cinema";
    hall.totalSeats = 50;
    hall.bookedSeats = 0;

    int choice, row, col;

    do {
        cout << "\n--- Cinema Hall Menu ---\n";
        cout << "1. Initialize Seats\n";
        cout << "2. Book a Seat\n";
        cout << "3. Cancel Booking\n";
        cout << "4. Display Seats\n";
        cout << "5. Exit\n";
        cout << "Enter your choice: ";
        cin >> choice;

        switch (choice) {
```

```
            case 1:
                initializeSeats(hall);
                break;
            case 2:
                cout << "Enter row (0-4) and column (0-9): ";
                cin >> row >> col;
                bookSeat(hall, row, col);
                break;
            case 3:
                cout << "Enter row (0-4) and column (0-9): ";
                cin >> row >> col;
                cancelSeat(hall, row, col);
                break;
            case 4:
                displaySeats(hall);
                break;
            case 5:
                cout << "Exiting...\n";
                break;
            default:
                cout << "Invalid choice!\n";
        }
    } while (choice != 5);

    return 0;
}
```

SAMPLE OUTPUT:

```
--- Cinema Hall Menu ---
1. Initialize Seats
2. Book a Seat
3. Cancel Booking
4. Display Seats
5. Exit
Enter your choice: 1

--- Cinema Hall Menu ---
1. Initialize Seats
2. Book a Seat
3. Cancel Booking
4. Display Seats
5. Exit
Enter your choice: 2
Enter row (0-4) and column (0-9): 2 3
Seat booked successfully!
```

--- Cinema Hall Menu ---
1. Initialize Seats
2. Book a Seat
3. Cancel Booking
4. Display Seats
5. Exit
Enter your choice: 2
Enter row (0-4) and column (0-9): 2 3
Seat already booked!

--- Cinema Hall Menu ---
1. Initialize Seats
2. Book a Seat
3. Cancel Booking
4. Display Seats
5. Exit
Enter your choice: 4

Seating Arrangement
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 1 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
Total Seats: 50 | Booked: 1 | Available: 49

--- Cinema Hall Menu ---
1. Initialize Seats
2. Book a Seat
3. Cancel Booking
4. Display Seats
5. Exit
Enter your choice: 3
Enter row (0-4) and column (0-9): 2 3
Booking cancelled!

--- Cinema Hall Menu ---
1. Initialize Seats
2. Book a Seat
3. Cancel Booking
4. Display Seats
5. Exit
Enter your choice: 5
Exiting...

Q9. Design a simple banking system using classes and encapsulation in C++.
Create a class bankAcccount with the following private data members:

1. **accountNumber**
2. **accountHolderName**
3. **balance**

Implement public methods to provide controlled access.

1. **Set AC Details: In Ac Details, set account number, name and balance**
2. **Deposit:**
3. **Withdraw**
4. **getBalance**
5. **Display Account Info**

Ensure data hiding, by keeping sensitive details like balance, name private and only accessible through methods.
Create multiple accounts using an array of objects, allow user to perform operations like deposit, withdraw, and show balance on these accounts.
Do not use constructors, instead initialize data through setter functions.

```
#include <iostream>
#include <string>
using namespace std;

class bankAccount {
private:
   int accountNumber;
   string accountHolderName;
   double balance;

public:
   void setAccountDetails(int no, string name, double bal) {
      accountNumber = no;
      accountHolderName = name;
      balance = bal;
   }

   void deposit(double amount) {
      if (amount > 0) {
         balance += amount;
         cout << "Amount deposited successfully.\n";
      } else {
         cout << "Enter a valid amount.\n";
      }
   }

   void withdraw(double amount) {
      if (amount > 0 && amount <= balance) {
         balance -= amount;
         cout << "Amount withdrawn successfully.\n";
      } else {
         cout << "Insufficient balance or invalid amount.\n";
```

```
        }
    }

    void displayAccountInfo(int no) {
        if (no == accountNumber) {
            cout << "---------------------------------\n";
            cout << "Account Number: " << accountNumber << endl;
            cout << "Account Holder Name: " << accountHolderName << endl;
            cout << "Available Balance: " << balance << endl;
            cout << "---------------------------------\n";
        } else {
            cout << "Invalid Account Number.\n";
        }
    }

    double getBalance(int no) {
        if (no == accountNumber) {
            return balance;
        } else {
            cout << "Invalid Account Number.\n";
            return -1;
        }
    }

    int getAccountNumber() {
        return accountNumber;
    }
};

int main() {
    bankAccount accounts[5];
    int totalAccounts = 0;

    int choice;
    do {
        cout << "\n========= Banking System Menu =========\n";
        cout << "1. Create Account\n";
        cout << "2. Deposit\n";
        cout << "3. Withdraw\n";
        cout << "4. Show Balance\n";
        cout << "5. Display Account Info\n";
        cout << "6. Exit\n";
        cout << "Enter your choice: ";
        cin >> choice;

        int accNo;
        string name;
        double bal, amount;
        bool found = false;
```

```
switch (choice) {
case 1:
    if (totalAccounts < 5) {
        cout << "Enter account number: ";
        cin >> accNo;
        cout << "Enter account holder name: ";
        cin.ignore();
        getline(cin, name);
        cout << "Enter initial balance: ";
        cin >> bal;
        accounts[totalAccounts].setAccountDetails(accNo, name, bal);
        totalAccounts++;
        cout << "Account created successfully!\n";
    } else {
        cout << "Account limit reached (5 accounts max).\n";
    }
    break;

case 2:
    cout << "Enter account number: ";
    cin >> accNo;
    cout << "Enter amount to deposit: ";
    cin >> amount;
    found = false;
    for (int i = 0; i < totalAccounts; i++) {
        if (accounts[i].getAccountNumber() == accNo) {
            accounts[i].deposit(amount);
            found = true;
            break;
        }
    }
    if (!found) cout << "Account not found.\n";
    break;

case 3:
    cout << "Enter account number: ";
    cin >> accNo;
    cout << "Enter amount to withdraw: ";
    cin >> amount;
    found = false;
    for (int i = 0; i < totalAccounts; i++) {
        if (accounts[i].getAccountNumber() == accNo) {
            accounts[i].withdraw(amount);
            found = true;
            break;
        }
    }
    if (!found) cout << "Account not found.\n";
```

```
            break;

        case 4:
            cout << "Enter account number: ";
            cin >> accNo;
            found = false;
            for (int i = 0; i < totalAccounts; i++) {
                if (accounts[i].getAccountNumber() == accNo) {
                    cout << "Balance: " << accounts[i].getBalance(accNo) << endl;
                    found = true;
                    break;
                }
            }
            if (!found) cout << "Account not found.\n";
            break;

        case 5:
            cout << "Enter account number: ";
            cin >> accNo;
            found = false;
            for (int i = 0; i < totalAccounts; i++) {
                accounts[i].displayAccountInfo(accNo);
                found = true;
                break;
            }
            if (!found) cout << "Account not found.\n";
            break;

        case 6:
            cout << "Exiting... Thank you for using the banking system!\n";
            break;

        default:
            cout << "Invalid choice! Please try again.\n";
        }
    } while (choice != 6);

    return 0;
}
```

SAMPLE OUTPUT:

========= Banking System Menu =========
1. Create Account
2. Deposit
3. Withdraw
4. Show Balance
5. Display Account Info
6. Exit
Enter your choice: 1
Enter account number: 101
Enter account holder name: Pawan Joshi
Enter initial balance: 5000
Account created successfully!

========= Banking System Menu =========
1. Create Account
2. Deposit
3. Withdraw
4. Show Balance
5. Display Account Info
6. Exit
Enter your choice: 2
Enter account number: 101
Enter amount to deposit: 2000
Amount deposited successfully.

========= Banking System Menu =========
1. Create Account
2. Deposit
3. Withdraw
4. Show Balance
5. Display Account Info
6. Exit
Enter your choice: 3
Enter account number: 101
Enter amount to withdraw: 1000
Amount withdrawn successfully.

========= Banking System Menu =========
1. Create Account
2. Deposit
3. Withdraw
4. Show Balance
5. Display Account Info
6. Exit
Enter your choice: 4
Enter account number: 101
Balance: 6000

========= Banking System Menu =========
1. Create Account
2. Deposit
3. Withdraw
4. Show Balance
5. Display Account Info
6. Exit
Enter your choice: 5
Enter account number: 101

Account Number: 101
Account Holder Name: Pawan Joshi
Available Balance: 6000

========= Banking System Menu =========
1. Create Account
2. Deposit
3. Withdraw
4. Show Balance
5. Display Account Info
6. Exit
Enter your choice: 6
Exiting... Thank you for using the banking system!

Q10. Define a class Hotel in C++ with the following specifications:
Private Members

- **Rno – Data member to store room number.**
- **Name – Data member to store customer name.**
- **Tariff – Data member to store charges per day.**
- **NOD – Data member to store number of days of stay.**
- **CALC() – Member function to calculate and return the total amount as:**
  - **Amount = NOD * Tariff**
  - **If Amount > 10000, then Amount = 1.05 * NOD * Tariff (5% extra charge).**

Public Members

- **Checkin() – Function to enter the values of Rno, Name, Tariff, and NOD.**
- **Checkout() – Function to display Rno, Name, Tariff, NOD, and the total amount (calculated using CALC() function).**

```
#include <iostream>
#include <string>
using namespace std;

class Hotel {
private:
   int Rno;
   string Name;
   double Tariff;
   int NOD;

   double CALC() {
      double amount = NOD * Tariff;
      if (amount > 10000) {
         amount = 1.05 * amount;
      }
      return amount;
   }

public:
   void Checkin() {
      cout << "Enter Room Number: ";
      cin >> Rno;
      cin.ignore();
      cout << "Enter Customer Name: ";
      getline(cin, Name);
      cout << "Enter Tariff per day: ";
      cin >> Tariff;
      cout << "Enter Number of Days: ";
      cin >> NOD;
   }

   void Checkout() {
      cout << "\n------ Bill Details ------\n";
      cout << "Room Number   : " << Rno << endl;
```

```
        cout << "Customer Name : " << Name << endl;
        cout << "Tariff per Day: " << Tariff << endl;
        cout << "Days Stayed   : " << NOD << endl;
        cout << "Total Amount  : " << CALC() << endl;
        cout << "-------------------------\n";
    }
};

int main() {
    Hotel h;
    h.Checkin();
    h.Checkout();
    return 0;
}
```

SAMPLE OUTPUT:

Case 1: Amount ≤ 10000

```
Enter Room Number: 101
Enter Customer Name: Pawan Joshi
Enter Tariff per day: 2000
Enter Number of Days: 4

------ Bill Details ------
Room Number   : 101
Customer Name : Pawan Joshi
Tariff per Day: 2000
Days Stayed   : 4
Total Amount  : 8000
-------------------------
```

Case 2: Amount > 10000 (5% extra applied)

```
Enter Room Number: 202
Enter Customer Name: Raj Sharma
Enter Tariff per day: 3000
Enter Number of Days: 4

------ Bill Details ------
Room Number   : 202
Customer Name : Raj Sharma
Tariff per Day: 3000
Days Stayed   : 4
Total Amount  : 12600
-------------------------
```

Q11. Imagine a tollbooth with a class called TollBooth. The two data items are of type unsigned int and double to hold the total number of cars and total amount of money collected. A constructor initializes both of these data members to 0. A member function called payingCar( )increments the car total and adds 0.5 to the cash total. Another function called nonPayCar( ) increments the car total but adds nothing to the cash total. Finally a member function called display( )shows the two totals. Include a program to test this class. This program should allow the user to push one key to count a paying car and another to count a non paying car. Pushing the ESC key should cause the program to print out the total number of cars and total cash and then exit.

```
#include <iostream>
#include <conio.h>
using namespace std;

class TollBooth{
  private:
    unsigned int totalCars;
    double totalAmount;

    public:
    TollBooth(){
      totalCars=0;
      totalAmount=0;
    }

    void payingCar(){
      totalAmount+=0.5;
      totalCars++;
    }

    void NonPayCar(){
      totalCars++;
    }

    void display(){
      cout<<"Total Cars:"<<totalCars<<endl;
      cout<<"Total Amount:"<<totalAmount<<endl;

    }

};

int main(){

  TollBooth t1;
  char ch;

  while(true){
    cout<<"Enter p for paying , n for non paying car and esc to End"<<endl;
```

```
    cout<<endl;

    ch=_getch();
    if (ch==27){
      t1.display();
      break;
    }

   if (ch=='p' ){
      t1.payingCar();
      cout<<"PAYING CAR ADDED\n"<<endl;
    }
    else if(ch=='n' ){
      t1.NonPayCar();
    cout<<"NON PAYING CAR ADDED\n"<<endl;
    }
  }
}
```

SAMPLE OUTPUT:

```
Enter p for paying , n for non paying car and esc to End

p
PAYING CAR ADDED

Enter p for paying , n for non paying car and esc to End

n
NON PAYING CAR ADDED

Enter p for paying , n for non paying car and esc to End

p
PAYING CAR ADDED

Enter p for paying , n for non paying car and esc to End

p
PAYING CAR ADDED

Enter p for paying , n for non paying car and esc to End

<ESC pressed>
Total Cars:4
Total Amount:1.5
```

Q12. Create a class called Time that has separate int member data for hours, minutes and seconds. One constructor should initialize this data to 0, and another should initialize it to fixed values. A member function should display it in 11:59:59 format. A member function named add() should add two objects of type time passed as arguments. A main ( ) program should create two initialized values together, leaving the result in the third time variable. Finally it should display the value of this third variable.

```
#include <iostream>

using namespace std;

class Time{
  private:
    int hours;
    int minutes;
    int seconds;

  public:
    Time(){
      hours=0;
      minutes=0;
      seconds=0;
    }

    Time(int h, int m, int s){
      hours=h;
      minutes=m;
      seconds=s;
    }

    void add(Time t11, Time t22){
      hours= t11.hours+ t22.hours;
      minutes= t11.minutes+ t22.minutes;
      seconds= t11.seconds+ t22.seconds;

      if(hours>12){
        hours%=12;
        minutes%=60;
        seconds%=60;
      }
    }

    void display(){
      cout<<hours<<" : "<< minutes <<" : " <<seconds <<endl;
    }

};

int main(){
  Time t1(12,12,12), t2(1, 1, 1),t3;
```

```
  t3.add(t1, t2);
  t3.display();
  return 0;
}
```

SAMPLE OUTPUT:

1 : 13 : 13

Q13. Create class SavingsAccount. Use a static variable annualInterestRate to store the annual interest rate for all account holders. Each object of the class contains a private instance variable savingsBalance indicating the amount the saver currently has on deposit. Provide method calculateMonthlyInterest() to calculate the monthly interest by multiplying the savingsBalance by annualInterestRate divided by 12.This interest should be added tosavingsBalance. Provide a static method modifyInterestRate() that sets the annualInterestRate to a new value. Write a program to test class SavingsAccount. Instantiate two savingsAccount objects, saver1 and saver2, with balances of Rs2000.00 and Rs3000.00, respectively. Set annualInterestRate to 4%, then calculate the monthly interest and print the new balances for both savers. Then set the annualInterestRate to 5%, calculate the next month's interest and print the new balances for both savers.

```
#include <iostream>
using namespace std;

class SavingsAccount {
private:
   double savingsBalance;
   static double annualInterestRate;

public:
   SavingsAccount(double balance = 0.0) {
      savingsBalance = balance;
   }

   void calculateMonthlyInterest() {
      double monthlyInterest = (savingsBalance * annualInterestRate) / 12;
      savingsBalance += monthlyInterest;
   }

   static void modifyInterestRate(double newRate) {
      annualInterestRate = newRate;
   }

   void displayBalance() const {
      cout << "Balance: Rs " << savingsBalance << endl;
   }
};

double SavingsAccount::annualInterestRate = 0.0;

int main() {
   SavingsAccount saver1(2000.00);
   SavingsAccount saver2(3000.00);

   SavingsAccount::modifyInterestRate(0.04);

   cout << "After setting interest rate to 4% and adding monthly interest:" << endl;
   saver1.calculateMonthlyInterest();
   saver2.calculateMonthlyInterest();
   cout << "Saver1 "; saver1.displayBalance();
   cout << "Saver2 "; saver2.displayBalance();
```

```
    cout << endl;

    SavingsAccount::modifyInterestRate(0.05);

    cout << "After setting interest rate to 5% and adding next month's interest:" << endl;
    saver1.calculateMonthlyInterest();
    saver2.calculateMonthlyInterest();
    cout << "Saver1 "; saver1.displayBalance();
    cout << "Saver2 "; saver2.displayBalance();

    return 0;
}
```

SAMPLE OUTPUT:

```
After setting interest rate to 4% and adding monthly interest:
Saver1 Balance: Rs 2006.67
Saver2 Balance: Rs 3010

After setting interest rate to 5% and adding next month's interest:
Saver1 Balance: Rs 2015.03
Saver2 Balance: Rs 3022.54
```

Qno.14: Create a class Complex with private data members real and imag representing the real and imaginary parts of a complex number.
Overload the + operator to add two complex numbers.
Input two complex numbers from the user and display their sum.

```
#include <iostream>
using namespace std;

class Complex {
private:
   float real, imag;

public:
   Complex(float r = 0, float i = 0) {
      real = r;
      imag = i;
   }

   Complex operator+(Complex c) {
      Complex temp;
      temp.real = real + c.real;
      temp.imag = imag + c.imag;
      return temp;
   }

   void input() {
      cout << "Enter real part: ";
      cin >> real;
      cout << "Enter imaginary part: ";
      cin >> imag;
   }

   void display() {
      if (imag >= 0)
         cout << real << " + " << imag << "i" << endl;
      else
         cout << real << " - " << -imag << "i" << endl;
   }
};

int main() {
   Complex c1, c2, c3;

   cout << "Enter first complex number:\n";
   c1.input();

   cout << "Enter second complex number:\n";
   c2.input();

   c3 = c1 + c2;

   cout << "\nSum of two complex numbers: ";
```

```
    c3.display();

    return 0;
}
```

SAMPLE OUTPUT:

```
Enter first complex number:
Enter real part: 3
Enter imaginary part: 4

Enter second complex number:
Enter real part: 5
Enter imaginary part: -2

Sum of two complex numbers: 8 + 2i
```

Qno.15: Create a class complex with real and imaginary parts overload+operator to add two complex number. Input two complex numbers and display their sum. Do this question using friend function.

```
#include <iostream>
using namespace std;

class Complex {
private:
   float real, imag;

public:
   Complex(float r = 0, float i = 0) {
      real = r;
      imag = i;
   }

   void input() {
      cout << "Enter real part: ";
      cin >> real;
      cout << "Enter imaginary part: ";
      cin >> imag;
   }

   void display() {
      if (imag >= 0)
         cout << real << " + " << imag << "i" << endl;
      else
         cout << real << " - " << -imag << "i" << endl;
   }

   friend Complex operator+(Complex c1, Complex c2);
};

Complex operator+(Complex c1, Complex c2) {
   Complex temp;
   temp.real = c1.real + c2.real;
   temp.imag = c1.imag + c2.imag;
   return temp;
}

int main() {
   Complex c1, c2, c3;

   cout << "Enter first complex number:\n";
   c1.input();

   cout << "Enter second complex number:\n";
   c2.input();

   c3 = c1 + c2;

   cout << "\nSum of two complex numbers: ";
```

```
    c3.display();

    return 0;
}
```

SAMPLE OUTPUT:

```
Enter first complex number:
Enter real part: 3
Enter imaginary part: 4
Enter second complex number:
Enter real part: 5
Enter imaginary part: -2

Sum of two complex numbers: 8 + 2i
```

Qno.16: Create a class matrix 3*3 +operator to add two matrices, -operator to substract two matrices and *to multiply to operator

```
#include <iostream>
using namespace std;

class Matrix {
private:
   int mat[3][3];

public:
   void input() {
      cout << "Enter 3x3 matrix elements:\n";
      for (int i = 0; i < 3; i++)
         for (int j = 0; j < 3; j++)
            cin >> mat[i][j];
   }

   void display() {
      for (int i = 0; i < 3; i++) {
         for (int j = 0; j < 3; j++)
            cout << mat[i][j] << "\t";
         cout << endl;
      }
   }

   Matrix operator+(Matrix m) {
      Matrix temp;
      for (int i = 0; i < 3; i++)
         for (int j = 0; j < 3; j++)
            temp.mat[i][j] = mat[i][j] + m.mat[i][j];
      return temp;
   }

   Matrix operator-(Matrix m) {
      Matrix temp;
      for (int i = 0; i < 3; i++)
         for (int j = 0; j < 3; j++)
            temp.mat[i][j] = mat[i][j] - m.mat[i][j];
      return temp;
   }

   Matrix operator*(Matrix m) {
      Matrix temp;
      for (int i = 0; i < 3; i++) {
         for (int j = 0; j < 3; j++) {
            temp.mat[i][j] = 0;
            for (int k = 0; k < 3; k++)
               temp.mat[i][j] += mat[i][k] * m.mat[k][j];
         }
      }
      return temp;
   }
```

```
};

int main() {
    Matrix m1, m2, sum, diff, prod;

    cout << "Enter first matrix:\n";
    m1.input();

    cout << "Enter second matrix:\n";
    m2.input();

    sum = m1 + m2;
    diff = m1 - m2;
    prod = m1 * m2;

    cout << "\nSum of matrices:\n";
    sum.display();

    cout << "\nDifference of matrices:\n";
    diff.display();

    cout << "\nProduct of matrices:\n";
    prod.display();

    return 0;
}
```

SAMPLE OUTPUT:

```
Enter first matrix:
1 2 3
4 5 6
7 8 9

Enter second matrix:
9 8 7
6 5 4
3 2 1

Sum of matrices:
10      10      10
10      10      10
10      10      10

Difference of matrices:
-8      -6      -4
-2       0       2
4        6       8

Product of matrices:
30      24      18
84      69      54
138     114     90
```