SOURCE CODE

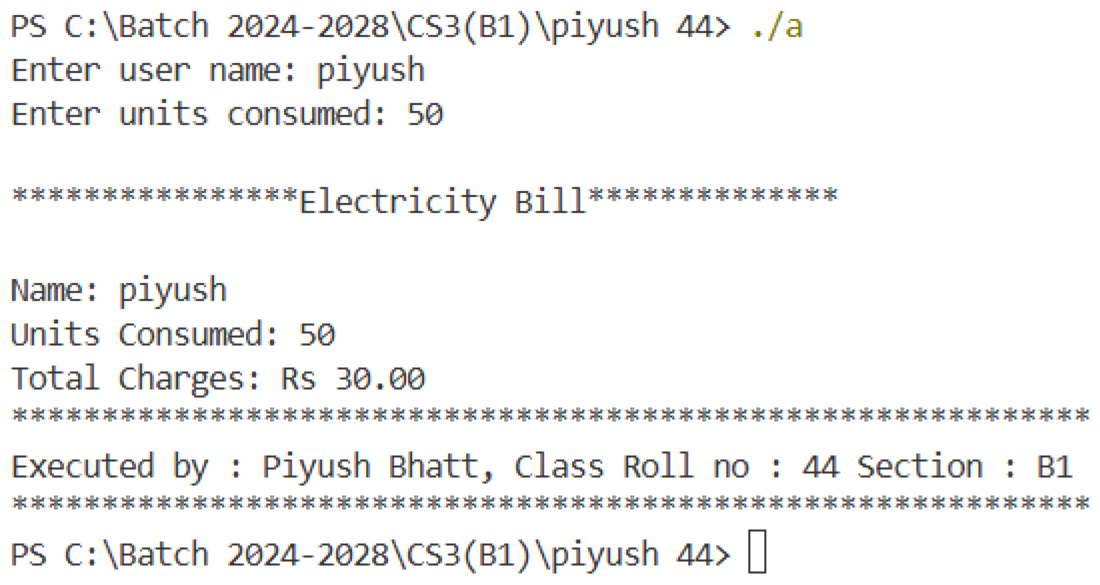
Q1. An electricity board charges the following rates to domestic users to discourage large consumption of energy.
For the first 100 units: - 60 P per unit
For the next 200 units: -80 P per unit
Beyond 300 units: -90 P per unit
All users are charged a minimum of Rs 50 if the total amount is more than Rs 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.

```
#include <iostream>
#include <iomanip>
#include <string>
using namespace std;
float calculateBill(int units){
   float bill=0.0;
   if(bill<50.0){
      bill=50.0;
   }
   if(units<=100){
      bill=units*0.60;
   }
   else if(units<=300){
      bill=100*0.60+(units-100)*0.80;
   }
   else{
      bill=100*0.60+200*0.80+(units-300)*0.90;
   }
   if(bill>300.0){
      bill+=bill*0.15;

   }
   return bill;
}
int main() {
   string name;
   int units;
   cout << "Enter user name: ";
   getline(cin, name);
   cout << "Enter units consumed: ";
   cin >> units;
   float totalBill = calculateBill(units);
   cout << fixed << setprecision(2);
```

```
    cout << "\n****************Electricity Bill**************\n"<<endl; cout
<< "Name: " << name << endl;
      cout << "Units Consumed: " << units << endl;
      cout << "Total Charges: Rs " << totalBill << endl;
      cout << "**********************************************************" << endl;
      cout << "Executed by: Piyush Bhatt, Class Roll no: 44 Section : B1" << endl;
      cout << "**********************************************************" << endl;
      return 0;
    }
```

OUTPUT

Q1.



```
PS C:\Batch 2024-2028\CS3(B1)\piyush 44> ./a
Enter user name: piyush
Enter units consumed: 50

****************Electricity Bill**************

Name: piyush
Units Consumed: 50
Total Charges: Rs 30.00
************************************************************
Executed by : Piyush Bhatt, Class Roll no : 44 Section : B1
************************************************************
PS C:\Batch 2024-2028\CS3(B1)\piyush 44> 
```

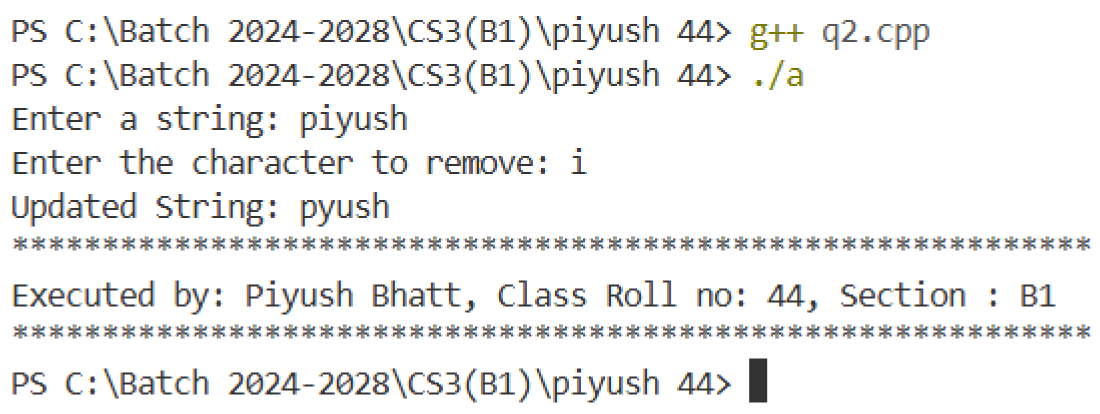
SOURCE CODE

Q2. Construct a C++ program that removes a specific character from a given string and return the updated string.
Typical Input: computer science is the future
Typical Output: compuer science is he fuure

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
   string str;
   char removecharacter;
   string result = "";
   cout << "Enter a string: ";
   getline(cin, str);
   cout << "Enter the character to remove: ";
   cin >> removecharacter;
   for (char c : str)
   {
      if (c != removecharacter)
      {
         result += c;
      }
   }
   cout << "Updated String: " << result << endl;
   cout << "*********************************************************" << endl;
   cout << “Executed by: Piyush Bhatt, Class Roll no: 44, Section : B1”  << endl;
   cout << "*********************************************************" << endl;
   return 0;
}
```

OUTPUT

Q2.



```
PS C:\Batch 2024-2028\CS3(B1)\piyush 44> g++ q2.cpp
PS C:\Batch 2024-2028\CS3(B1)\piyush 44> ./a
Enter a string: piyush
Enter the character to remove: i
Updated String: pyush
*************************************************************
Executed by: Piyush Bhatt, Class Roll no: 44, Section : B1
*************************************************************
PS C:\Batch 2024-2028\CS3(B1)\piyush 44> 
```

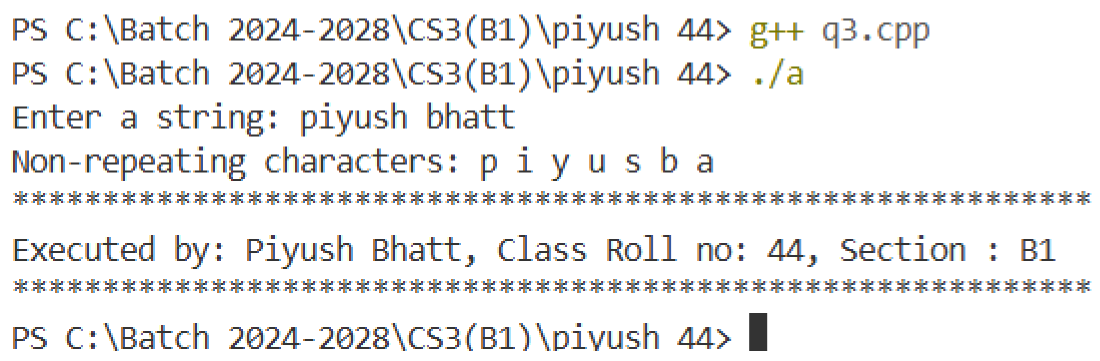
SOURCE CODE

Q3. Implement a C++ program to find the non-repeating characters in string.
Typical Input: graphic era university
Typical Output: c g h n p s t u v y

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
    string str;
    cout << "Enter a string: ";
    getline(cin, str);
    int freq[256] = {0};
    for (char c : str)
    {
        if (c != ' ')
        {
            freq[(unsigned char)c]++;
        }
    }
    cout << "Non-repeating characters: ";
    for (char c : str)
    {
        if (c != ' ' && freq[(unsigned char)c] == 1)
        {
            cout << c << " ";
        }
    }
    cout << endl;
    cout << "************************************************************" << endl;
    cout << “Executed by: Piyush Bhatt, Class Roll no: 44, Section : B1”  << endl;
    cout << "************************************************************" << endl;
    return 0;
}
```

OUTPUT

Q3.



```
PS C:\Batch 2024-2028\CS3(B1)\piyush 44> g++ q3.cpp
PS C:\Batch 2024-2028\CS3(B1)\piyush 44> ./a
Enter a string: piyush bhatt
Non-repeating characters: p i y u s b a
************************************************************
Executed by: Piyush Bhatt, Class Roll no: 44, Section : B1
************************************************************
PS C:\Batch 2024-2028\CS3(B1)\piyush 44> 
```

SOURCE CODE

Q4. 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 sum of all the other indices. To calculate the special sum for any index you pick the first element that is and add it to your sum. Now you pick next two elements i.e., and and add both of them to your sum. Now you will pick the next elements, and this continues till the index for which it is possible to pick the elements. Find the best index and in the output print its corresponding special sum. Note that there may be more than one best index, but you need to only print the maximum special sum.

Input

First line contains an integer as input. Next line contains space separated integers denoting the elements of the array

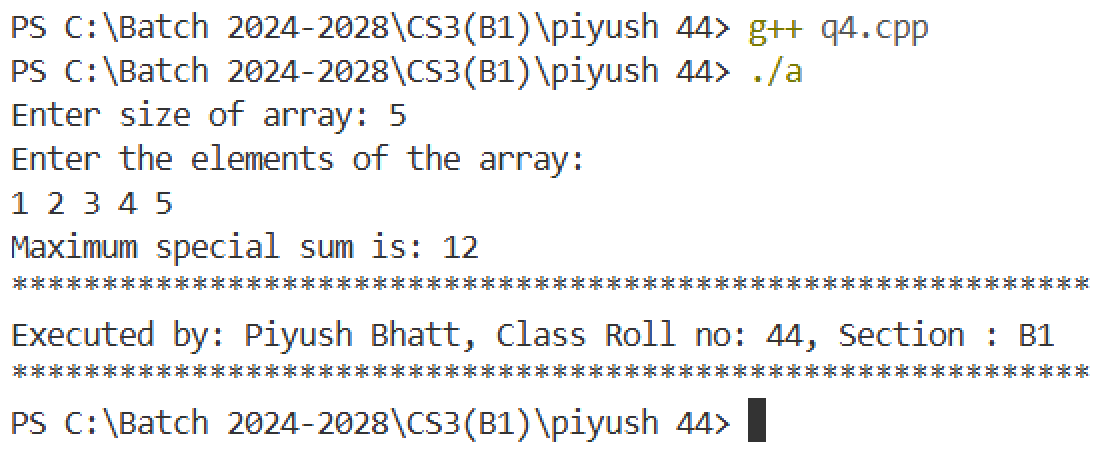
Output

In the output you have to print an integer that denotes the maximum special Sum.

```
#include <iostream>
#include <climits>
using namespace std;
int main()
{
   int i, j, n;
   cout << "Enter size of array: ";
   cin >> n;
   int arr[n];
   cout << "Enter the elements of the array:\n";
   for (i = 0; i < n; i++)
   {
      cin >> arr[i];
   }
   int max_sum = INT_MIN;
   for (i = 0; i < n; i++)
   {
      int group = 1;
      int sum = 0;
      int index = i;
      while (index + group <= n)
      {
         for (j = 0; j < group; j++)
         {
            sum += arr[index + j];
            }
            index += group;
            group++;
        }
        if (sum > max_sum)
            max_sum = sum;
    }
    cout << "Maximum special sum is: " << max_sum << endl;
    cout << "*************************************************************" << endl;
    cout << “Executed by: Piyush Bhatt, Class Roll no: 44, Section : B1”  << endl;
    cout << "*************************************************************" << endl;
    return 0;

}
```

OUTPUT

Q4.



```
PS C:\Batch 2024-2028\CS3(B1)\piyush 44> g++ q4.cpp
PS C:\Batch 2024-2028\CS3(B1)\piyush 44> ./a
Enter size of array: 5
Enter the elements of the array:
1 2 3 4 5
Maximum special sum is: 12
************************************************************
Executed by: Piyush Bhatt, Class Roll no: 44, Section : B1
************************************************************
PS C:\Batch 2024-2028\CS3(B1)\piyush 44> 
```

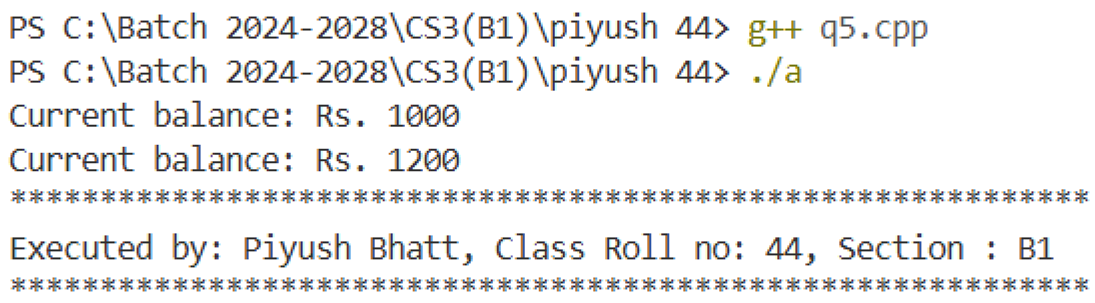
SOURCE CODE

Q5. Implement a C++ program to demonstrate the concept of data abstraction using the concept of Class and Objects

```
#include  <iostream>
using namespace std;
class Bank
{
private:
   double balance;
public:
   Bank(double initialBalance)
   {
      if (initialBalance >= 0)
         balance = initialBalance;
      else
         balance = 0;
   }
   void deposit(double amount)
   {
      if (amount > 0)
         balance += amount;
   }
   void withdraw(double amount)
   {
      if (amount > 0 && amount <= balance)
         balance -= amount;
      else
         cout << "Insufficient balance or invalid amount!" << endl;
   }
   double getBalance()
   {
      return balance;
   }
};
int main()
{
   Bank b1(1000);
   cout << "Current balance: Rs. " << b1.getBalance() << endl;
   b1.deposit(400);
   b1.withdraw(200);
   cout << "Current balance: Rs. " << b1.getBalance() << endl;
   cout << "**********************************************************" << endl;
   cout << “Executed by: Piyush Bhatt, Class Roll no: 44, Section : B1”  << endl;
   cout << "**********************************************************" << endl;
   return 0;}
```

OUTPUT

Q5.



```
PS C:\Batch 2024-2028\CS3(B1)\piyush 44> g++ q5.cpp
PS C:\Batch 2024-2028\CS3(B1)\piyush 44> ./a
Current balance: Rs. 1000
Current balance: Rs. 1200
************************************************************
Executed by: Piyush Bhatt, Class Roll no: 44, Section : B1
************************************************************
```

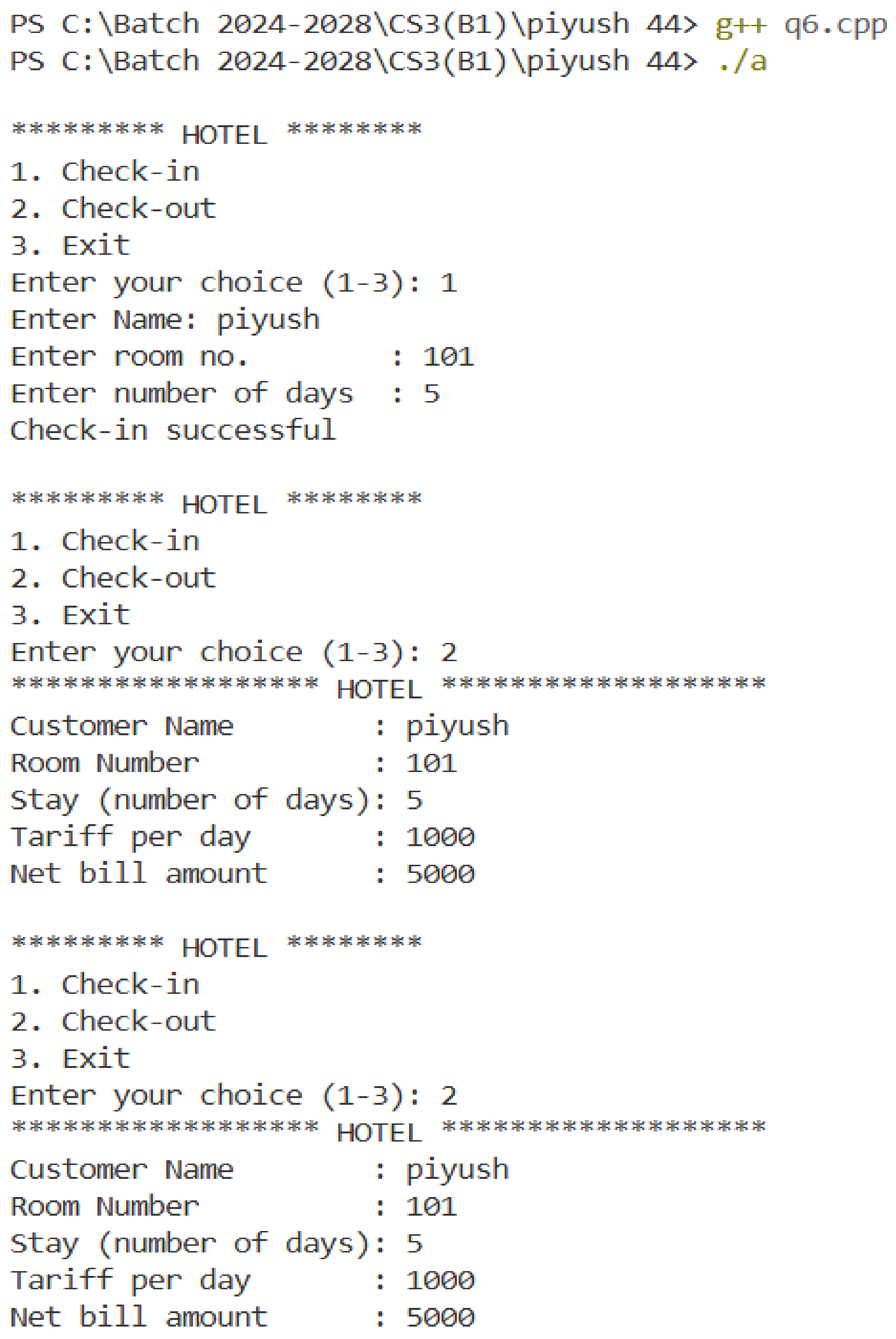
SOURCE CODE

Q6. 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 per day charges
• NOD Data member to store number of days of stay
• CALC() Function to calculate and return amount as NOD*Tariff ,and if the value of days* Tariff >10000,
then total amount is 1.05* days*Tariff.
Public members
• Checkin() Function to enter the content Rno, Name, Tariff and NOD
• Checkout() Function to display Rno, Name, Tariff,
NOD and Amount (amount to be displayed by calling function) CALC()

```
#include  <iostream>
using namespace std;
class hotel
{
private:
   int Rno;
   string name;
   const int tariff = 1000;
   int NOD;
   double amount;
   bool checkedIN = false;

public:
   void calc()
   {
      amount = NOD * tariff;
      if (amount > 10000)
         amount = 1.05 * NOD * tariff;
   }
   void checkin(string x, int r1, int days)
   {
      name = x;
      Rno = r1;
      NOD = days;
      checkedIN = true;
      cout << "checkin successful\n";
   }
   void checkout()
   {
        if (checkedIN)
        {
            cout << "******************HOTEL*******************" << endl;
            cout << "Custome Name : " << name << endl;
            cout << "Room Number : " << Rno << endl;
            cout << "Stay(number of days) : " << NOD << endl;
            cout << "Tariff per day : " << tariff << endl;
            cout << "Net bill amount " << amount << endl;
        }
        else{
            cout << "error" << endl;
            cout << "Please checkin first\n";
        }
    }
};
int main()
{
    string x;
    int room, days;
    hotel r1;
    int choice;
    do
    {
        cout << "\n********* HOTEL ********\n";
        cout << "1. Checkin\n";
        cout << "2. Checkout\n";
        cout << "3. Exit\n";
        cout << "Enter your choice (1-3): ";
        cin >> choice;
        switch (choice)
        {
        case 1:
            cout << "Enter Name: ";
            getchar();
            getline(cin, x);
            cout << "Enter room no. : ";
            cin >> room;
            cout << "Enter number of days : ";
            cin >> days;
            r1.checkin(x, room, days);
            r1.calc();
            break;
        case 2:
            r1.checkout();
            break;
        case 3:
            cout << "Thank you\n";
            break;
        default:
            cout << "Invalid choice!\n";
        }
    } while (choice != 3);
    cout << "**********************************************************" << endl;
    cout << “Executed by: Piyush Bhatt, Class Roll no: 44, Section : B1”  << endl;
    cout << "**********************************************************" << endl;
    return 0;
}
```

OUTPUT

Q6.



```
PS C:\Batch 2024-2028\CS3(B1)\piyush 44> g++ q6.cpp
PS C:\Batch 2024-2028\CS3(B1)\piyush 44> ./a

********* HOTEL ********
1. Check-in
2. Check-out
3. Exit
Enter your choice (1-3): 1
Enter Name: piyush
Enter room no.        : 101
Enter number of days  : 5
Check-in successful

********* HOTEL ********
1. Check-in
2. Check-out
3. Exit
Enter your choice (1-3): 2
****************** HOTEL *******************
Customer Name         : piyush
Room Number           : 101
Stay (number of days): 5
Tariff per day        : 1000
Net bill amount       : 5000

********* HOTEL ********
1. Check-in
2. Check-out
3. Exit
Enter your choice (1-3): 2
****************** HOTEL *******************
Customer Name         : piyush
Room Number           : 101
Stay (number of days): 5
Tariff per day        : 1000
Net bill amount       : 5000
```

SOURCE CODE

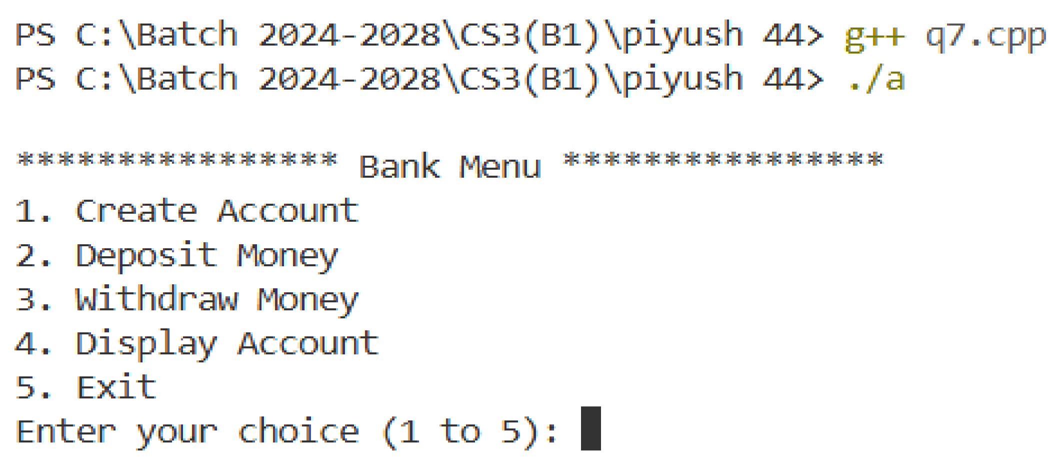
Q7. Implement a Program in C++ by defining a class to represent a bank account.
Include the following:
Data Members
- **Name of the depositor**
- **Account number**
- **Type of account (Saving, Current etc.)**
- **Balance amount in the account**

Member Functions
- **To assign initial values**
- **To deposit an amount**
- **To withdraw an amount after checking the balance**
- **To display name and balance**

```
#include  <iostream>
using namespace std;
class Bank
{
private:
    string name;
    string acc_no;
    string acc_type;
    int balance;

public:
    void createAccount()
    {
        cout << "Enter Account Holder Name: ";
        getchar();
        getline(cin, name);
        cout << "Enter Account Number: ";
        getline(cin, acc_no);
        cout << "Enter Account Type (Saving/Current): ";
        getline(cin, acc_type);
        cout << "Enter Initial Balance: ";
        cin >> balance;
        cout << "Account Created Successfully!\n";
    }
    void deposit(int amount)
    {
        if (amount > 0)
        {
            balance += amount;
            cout << "Amount deposited successfully.\n";
        }
        else
        {
            cout << "Invalid amount.\n";
        }
    }
    void withdraw(int amount)
    {
        if (amount <= balance && amount > 0)
        {
            balance -= amount;
            cout << "Amount withdrawn successfully.\n";
            cout << "Remaining balance : " << balance << endl;
        }
        else
        {
            cout << "Insufficient balance or invalid amount.\n";
        }
    }
    void display()
    {
        cout << "\nAccount Holder Name: " << name << endl;
        cout << "Account Number: " << acc_no << endl;
        cout << "Account Type: " << acc_type << endl;
        cout << "Balance: " << balance << endl;
    }
};
int main()
{
    Bank b;
    int choice;
    do
    {
        cout << "\n*************** Bank Menu ***************\n";
        cout << "1. Create Account\n";
        cout << "2. Deposit Money\n";
        cout << "3. Withdraw Money\n";
        cout << "4. Display Account\n";
        cout << "5. Exit\n";
        cout << "Enter your choice (1 to 5): ";
        cin >> choice;
        switch (choice)
        {
        case 1:
            b.createAccount();
            break;
        case 2:
            int amount;
            cout << "Enter amount to deposit: ";
            cin >> amount;
            b.deposit(amount);
            break;
        case 3:
            cout << "Enter amount to withdraw: ";
            cin >> amount;
            b.withdraw(amount);
            break;
        case 4:
            b.display();
            break;
        case 5:
            cout << "Thank you for using the bank system\n";
            break;
        default:
            cout << "Invalid choice! Please try again.\n";
        }
    } while (choice != 5);
    cout << "********************************************************" << endl;
    cout << “Executed by: Piyush Bhatt, Class Roll no: 44, Section : B1”  << endl;
    cout << "********************************************************" << endl;
    return 0;
}
```

OUTPUT

Q7



```
PS C:\Batch 2024-2028\CS3(B1)\piyush 44> g++ q7.cpp
PS C:\Batch 2024-2028\CS3(B1)\piyush 44> ./a

**************** Bank Menu ****************
1. Create Account
2. Deposit Money
3. Withdraw Money
4. Display Account
5. Exit
Enter your choice (1 to 5): 
```

SOURCE CODE

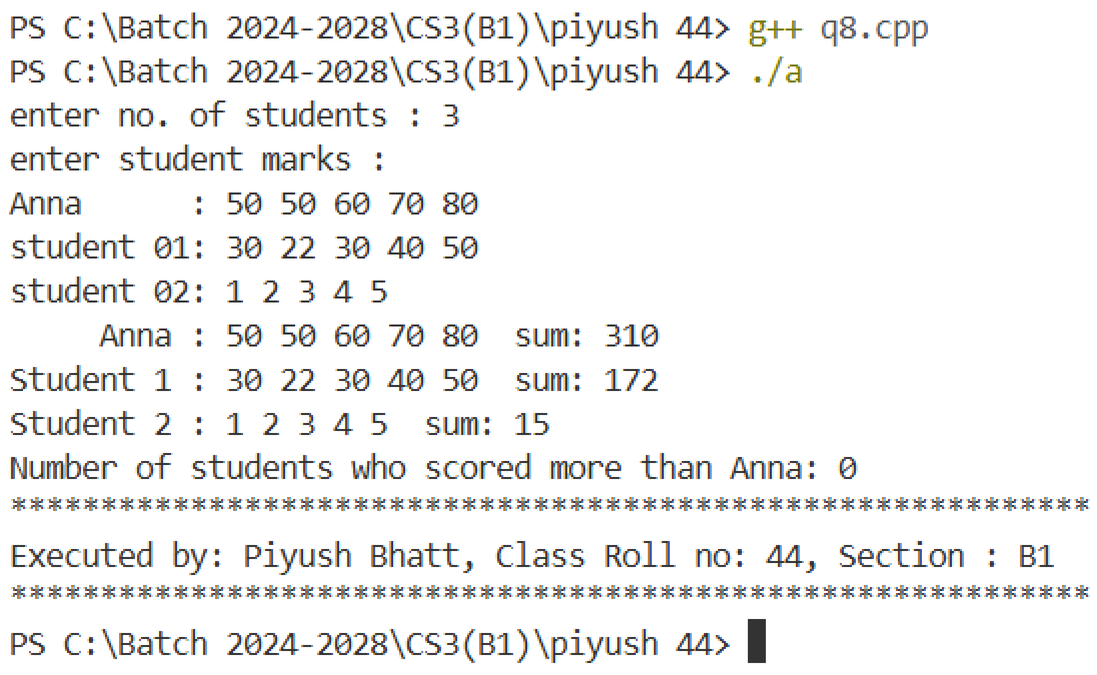
Q8. Anna is a contender for valedictorian of her high school. She wants to know how many students (if any) have scored higher than her in the exams given during this semester.
Create a class named Student with the following specifications:
Ø An instance variable named scores holds a student's 5 exam scores.
Ø A void input () function reads 5 integers and saves them to scores.
Ø An int calculateTotalScore() function that returns the sum of the student's scores.
Input Format
In the void Student::input() function, you must read 5 scores from standard input and save them to your scores instance variable.
Output Format
In the int Student::calculateTotalScore() function, you must return the student's total grade (the sum of the values in scores).
The code in the editor will determine how many scores are larger than Anna's and print that number to the console.
Sample Input
The first line contains n, the number of students in Anna's class. The n subsequent lines contain each student's 5 exam grades for this semester.

```
#include  <iostream>
#include  <iomanip>
using namespace std;
class student
{
private:
   int scores[5];

public:
   void input()
   {
      for (int i = 0; i < 5; i++)
      {
         cin >> scores[i];
      }
   }
   void display()
   {
      for (int i = 0; i < 5; i++)
      {
         cout << scores[i] << " ";
      }
   }
```

```
    int calculate()
    {
        int totalscore = 0;
        for (int i = 0; i < 5; i++)
        {
            totalscore += scores[i];
        }
        return totalscore;
    }
};
int main()
{
    int n;
    cout << "enter no. of students : ";
    cin >> n;
    student st[n];
    cout << "enter student marks :\n";
    for (int i = 0; i < n; i++)
    {
        if (i == 0)
        {
            cout << "Anna" << setw(8) << ": ";
            st[i].input();
        }
        else
        {
            cout << "student " << setw(2) << setfill('0') << i << ": ";
            st[i].input();
        }
    }
    for (int i = 0; i < n; i++)
    {
        if (i == 0)
        {
            cout << setw(12) << setfill(' ') << "Anna : ";
        }
        else
        {
            cout << "Student " << i << " : ";
        }
        st[i].display();
        cout << " sum: " << st[i].calculate() << endl;
    }
    int c = 0;
    int annaTotal = st[0].calculate();
    for (int i = 1; i < n; i++)
    {
        if (st[i].calculate() > annaTotal)
        {
           c++;
        }
    }
    cout << "Number of students who scored more than Anna: " << c << endl;
    cout << "****************************************************************" << endl;
    cout << “Executed by: Piyush Bhatt, Class Roll no: 44, Section : B1”  << endl;
    cout << "****************************************************************" << endl;
    return 0;
}
```

OUTPUT

Q8.



```
PS C:\Batch 2024-2028\CS3(B1)\piyush 44> g++ q8.cpp
PS C:\Batch 2024-2028\CS3(B1)\piyush 44> ./a
enter no. of students : 3
enter student marks :
Anna      : 50 50 60 70 80
student 01: 30 22 30 40 50
student 02: 1 2 3 4 5
     Anna : 50 50 60 70 80  sum: 310
Student 1 : 30 22 30 40 50  sum: 172
Student 2 : 1 2 3 4 5  sum: 15
Number of students who scored more than Anna: 0
************************************************************
Executed by: Piyush Bhatt, Class Roll no: 44, Section : B1
************************************************************
PS C:\Batch 2024-2028\CS3(B1)\piyush 44>
```

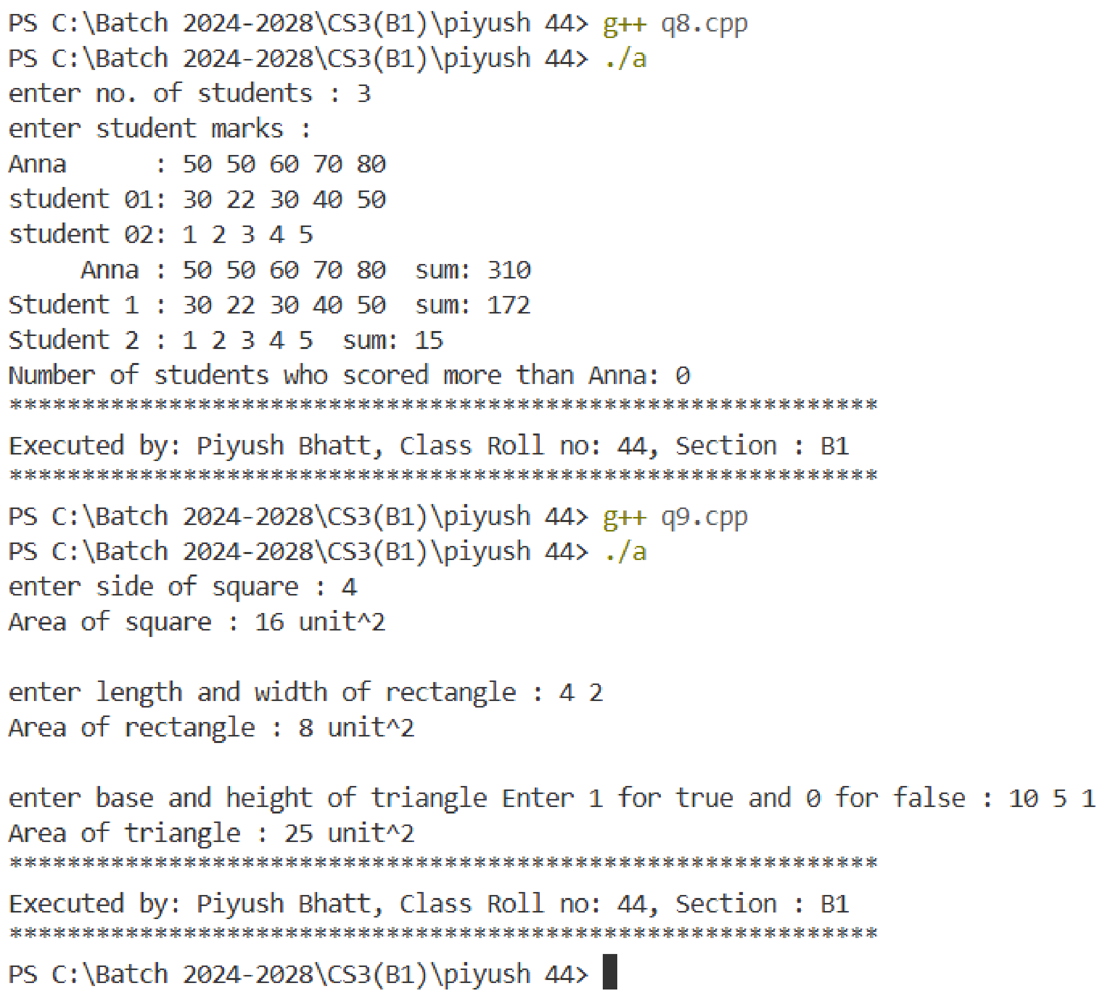
SOURCE CODE

Q9. Construct a Program in C++ to show the working of function overloading(compile time polymorphism) by using a function named calculate Area () to calculate area of square, rectangle and triangle using different signatures as required.

```
#include <iostream>
using namespace std;
double calculateArea(double side)
{
   return side * side;
}
double calculateArea(double length, double width)
{
   return length * width;
}
double calculateArea(double base, double height, bool isTriangle)
{
   if (isTriangle)
      return 0.5 * base * height;
   else
      return base * height;
}
int main()
{
   bool x;
   double side, length, width, base, height;
   cout << "enter side of square : ";
   cin >> side;
   cout << "Area of square : " << calculateArea(side) << " unit^2" << endl;
   cout << "\nenter length and width of rectangle : ";
   cin >> length >> width;
   cout << "Area of rectangle : " << calculateArea(length, width) << " unit^2" << endl;
   cout << "\nenter base and height of triangle Enter 1 for true and 0 for false : ";
   cin >> base >> height >> x;
   cout << "Area of triangle : " << calculateArea(base, height, x) << " unit^2" << endl;
   cout << "************************************************************" << endl;
   cout << "Executed by: Piyush Bhatt, Class Roll no: 44, Section : B1"  << endl;
   cout << "************************************************************" << endl;
   return 0;
}
```

OUTPUT

Q9.



```
PS C:\Batch 2024-2028\CS3(B1)\piyush 44> g++ q8.cpp
PS C:\Batch 2024-2028\CS3(B1)\piyush 44> ./a
enter no. of students : 3
enter student marks :
Anna      : 50 50 60 70 80
student 01: 30 22 30 40 50
student 02: 1 2 3 4 5
     Anna : 50 50 60 70 80  sum: 310
Student 1 : 30 22 30 40 50  sum: 172
Student 2 : 1 2 3 4 5  sum: 15
Number of students who scored more than Anna: 0
************************************************************
Executed by: Piyush Bhatt, Class Roll no: 44, Section : B1
************************************************************
PS C:\Batch 2024-2028\CS3(B1)\piyush 44> g++ q9.cpp
PS C:\Batch 2024-2028\CS3(B1)\piyush 44> ./a
enter side of square : 4
Area of square : 16 unit^2

enter length and width of rectangle : 4 2
Area of rectangle : 8 unit^2

enter base and height of triangle Enter 1 for true and 0 for false : 10 5 1
Area of triangle : 25 unit^2
************************************************************
Executed by: Piyush Bhatt, Class Roll no: 44, Section : B1
************************************************************
PS C:\Batch 2024-2028\CS3(B1)\piyush 44>
```

SOURCE CODE

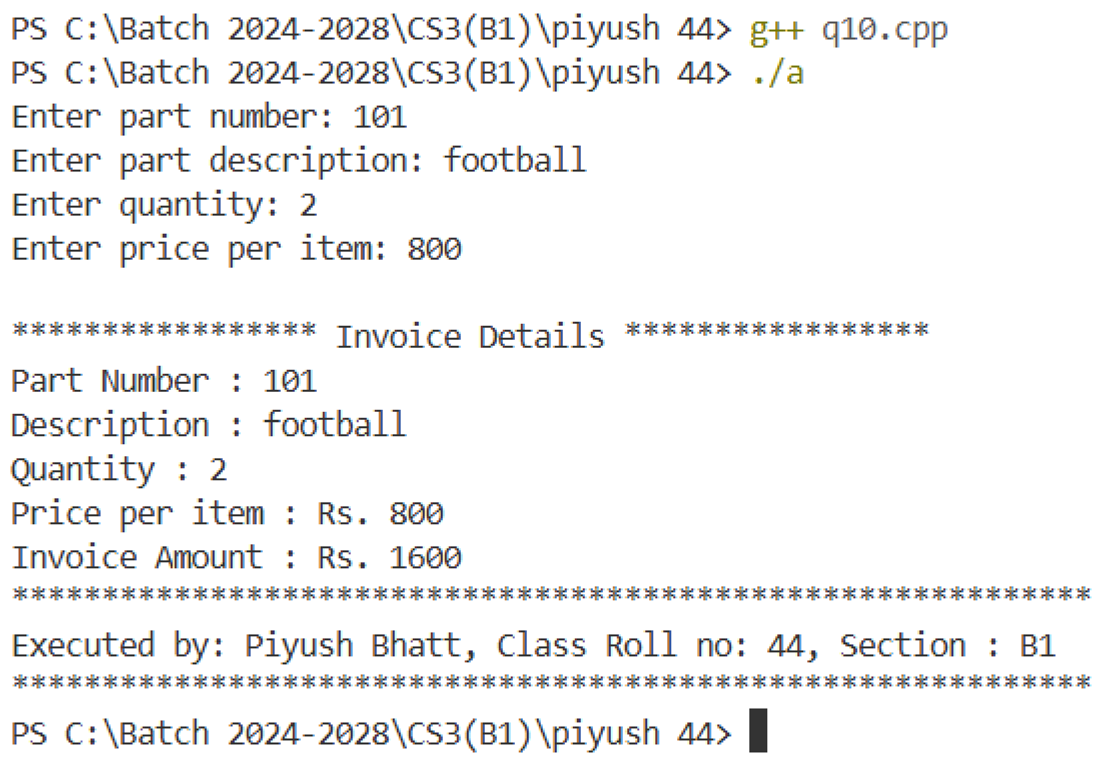
Q10. Create a class called Invoice that a hardware store might use to represent an invoice for an item sold at the store. An Invoice should include four pieces of information as instance.
Data Members -
- **partNumber (type String)**
- **partDescription (type String)**
- **quantity of the item being purchased (type int)**
- **price_per_item (type 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 getInvoiceAmount() 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 to0.0. Write a test application named invoiceTest that demonstrates class Invoice's capabilities.

```
#include <iostream>
#include <string>
using namespace std;
class Invoice
{
private:
    string partNumber;
    string partDescription;
    int quantity;
    double pricePerItem;

public:
    Invoice(string pNumber, string pDesc, int qty, double price)
    {
        partNumber = pNumber;
        partDescription = pDesc;
        setQuantity(qty);
        setPricePerItem(price);
    }
    void setPartNumber(string pNumber)
    {
        partNumber = pNumber;
    }
    void setPartDescription(string pDesc)
    {
        partDescription = pDesc;
    }
    void setQuantity(int qty)
    {
        if (qty > 0)
            quantity = qty;
        else
            quantity = 0;
    }
    void setPricePerItem(double price)
    {
        if (price > 0)
            pricePerItem = price;
        else
            pricePerItem = 0.0;
    }
    string getPartNumber()
    {
        return partNumber;
    }
    string getPartDescription()
    {
        return partDescription;
    }
    int getQuantity()
    {
        return quantity;
    }
    double getPricePerItem()
    {
        return pricePerItem;
    }
    double getInvoiceAmount()
    {
        return quantity * pricePerItem;
    }
};
int main()
{
    string partNum, partDesc;
    int qty;
    double price;
    cout << "Enter part number: ";
    getline(cin, partNum);
    cout << "Enter part description: ";
    getline(cin, partDesc);
    cout << "Enter quantity: ";
    cin >> qty;
    cout << "Enter price per item: ";
    cin >> price;
    Invoice item(partNum, partDesc, qty, price);
    cout << "\n***************** Invoice Details *****************" << endl;
    cout << "Part Number : " << item.getPartNumber() << endl;
    cout << "Description : " << item.getPartDescription() << endl;
    cout << "Quantity : " << item.getQuantity() << endl;
    cout << "Price per item : Rs. " << item.getPricePerItem() << endl;
    cout << "Invoice Amount : Rs. " << item.getInvoiceAmount() << endl;
    cout << "**********************************************************" << endl;
    cout << “Executed by: Piyush Bhatt, Class Roll no: 44, Section : B1”  << endl;
    cout << "**********************************************************" << endl;
    return 0;
}
```

OUTPUT

Q10.



```
PS C:\Batch 2024-2028\CS3(B1)\piyush 44> g++ q10.cpp
PS C:\Batch 2024-2028\CS3(B1)\piyush 44> ./a
Enter part number: 101
Enter part description: football
Enter quantity: 2
Enter price per item: 800

***************** Invoice Details *****************
Part Number : 101
Description : football
Quantity : 2
Price per item : Rs. 800
Invoice Amount : Rs. 1600
************************************************************
Executed by: Piyush Bhatt, Class Roll no: 44, Section : B1
************************************************************
PS C:\Batch 2024-2028\CS3(B1)\piyush 44>
```

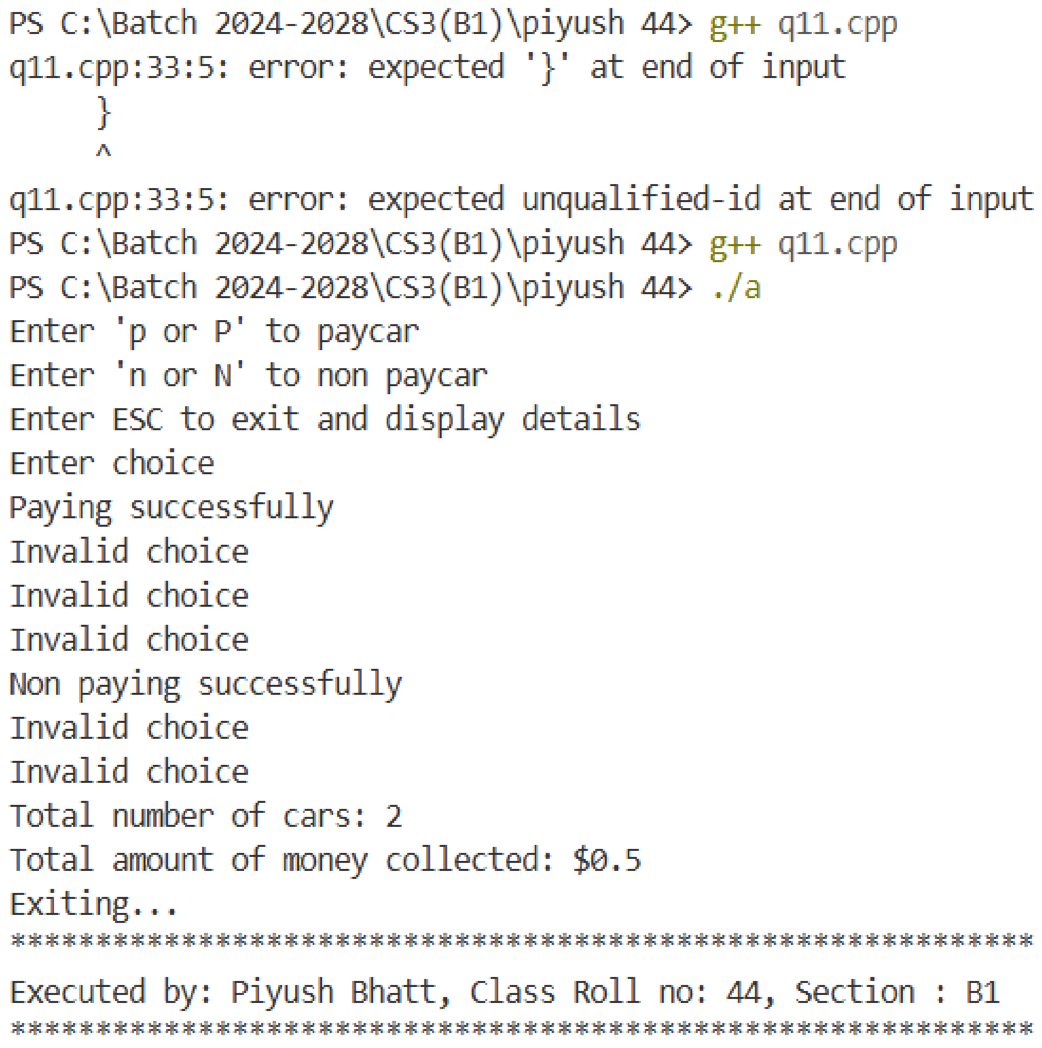
SOURCE CODE

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 cars;
   double money;

public:
   TollBooth()
   {
      cars = 0;
      money = 0.0;
   }
   void payingCar()
   {
      cars++;
      money += 0.5;
   }
   void nonPayCar()
   {
      cars++;
   }
   void display()
   {
      cout << "Total number of cars: " << cars << endl;
      cout << "Total amount of money collected: $" << money << endl;
   }
};
int main()
{
    TollBooth booth;
    char ch;
    cout << "Enter 'p or P' to paycar" << endl;
    cout << "Enter 'n or N' to non paycar" << endl;
    cout << "Enter ESC to exit and display details " << endl;
    cout << "enter choice\n";
    while (true)
    {
        ch = _getch();
        if (ch == 'p' || ch == 'P')
        {
            booth.payingCar();
            cout << "paying succesfully\n";
        }
        else if (ch == 'n' || ch == 'N')
        {
            booth.nonPayCar();
            cout << "Non paying successfully\n";
        }
        else if (ch == 27)
        {
            booth.display();
            cout << "Exiting...";
            break;
        }
        else
            cout << "Invalid choice\n";
    }
    cout << "*********************************************************" << endl;
    cout << “Executed by: Piyush Bhatt, Class Roll no: 44, Section : B1”  << endl;
    cout << "*********************************************************" << endl;
    return 0;
}
```

OUTPUT

Q11.



```
PS C:\Batch 2024-2028\CS3(B1)\piyush 44> g++ q11.cpp
q11.cpp:33:5: error: expected '}' at end of input
     }
     ^
q11.cpp:33:5: error: expected unqualified-id at end of input
PS C:\Batch 2024-2028\CS3(B1)\piyush 44> g++ q11.cpp
PS C:\Batch 2024-2028\CS3(B1)\piyush 44> ./a
Enter 'p or P' to paycar
Enter 'n or N' to non paycar
Enter ESC to exit and display details
Enter choice
Paying successfully
Invalid choice
Invalid choice
Invalid choice
Non paying successfully
Invalid choice
Invalid choice
Total number of cars: 2
Total amount of money collected: $0.5
Exiting...
************************************************************
Executed by: Piyush Bhatt, Class Roll no: 44, Section : B1
************************************************************
```

SOURCE CODE

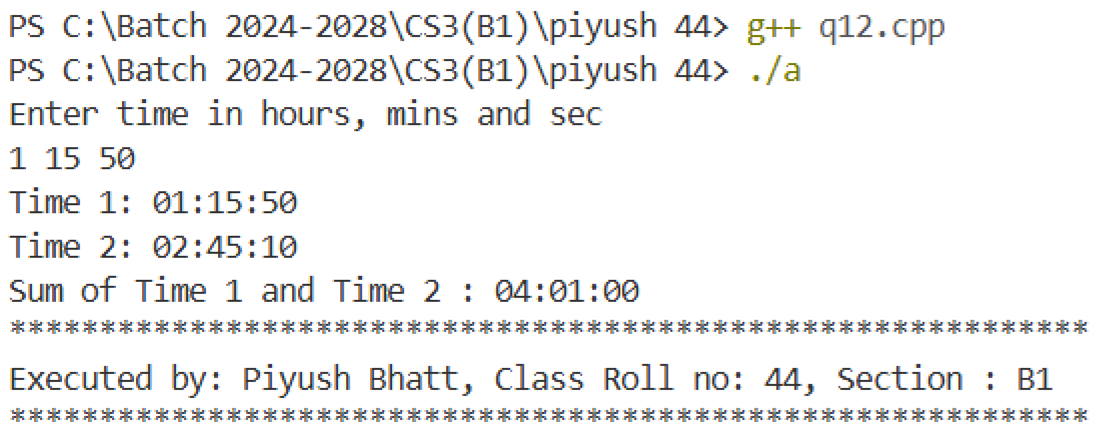
Q12. Create a class called Time that has separate int member data for hours, minutes and seconds. One constructor should initialize this data to 0, and another should initialize it to fixed values. A member function should display it in 11:59:59 format. A member function named add() should add two objects 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>
#include   <iomanip>
using namespace std;
class Time
{
private:
   int hr, min, sec;
public:
   Time(int h = 0, int m = 0, int st = 0) : hr(h), min(m), sec(st) {}
   void display()
   {
       cout << setw(2) << setfill('0') << hr << ":"
            << setw(2) << setfill('0') << min << ":"
           << setw(2) << setfill('0') << sec << endl;
   }
   void sum(Time t1, Time t2)
   {
      sec = t1.sec + t2.sec;
      min = t1.min + t2.min + sec/60;
      sec = sec % 60;
      hr = t1.hr + t2.hr + min/60;
      min = min % 60;
   }
};
int main()
{
   int h, m, st;
   cout << "Enter time in hours, mins and sec" << endl;
   cin >> h >> m >> st;
   Time t1(h, m, st);
   Time t2(2, 45, 10);
   Time t3;
   t3.sum(t1, t2);
   cout << "Time 1: ";
   t1.display();
   cout << "Time 2: ";
```

```
    t2.display();
    cout << "Sum of Time 1 and Time 2 : ";
    t3.display();
    cout << "*****************************************************************" << endl;
    cout << “Executed by: Piyush Bhatt, Class Roll no: 44, Section : B1”  << endl;
    cout << "*****************************************************************" << endl;
    return 0;
}
```

OUTPUT

Q12.



```
PS C:\Batch 2024-2028\CS3(B1)\piyush 44> g++ q12.cpp
PS C:\Batch 2024-2028\CS3(B1)\piyush 44> ./a
Enter time in hours, mins and sec
1 15 50
Time 1: 01:15:50
Time 2: 02:45:10
Sum of Time 1 and Time 2 : 04:01:00
************************************************************
Executed by: Piyush Bhatt, Class Roll no: 44, Section : B1
************************************************************
```

SOURCE CODE

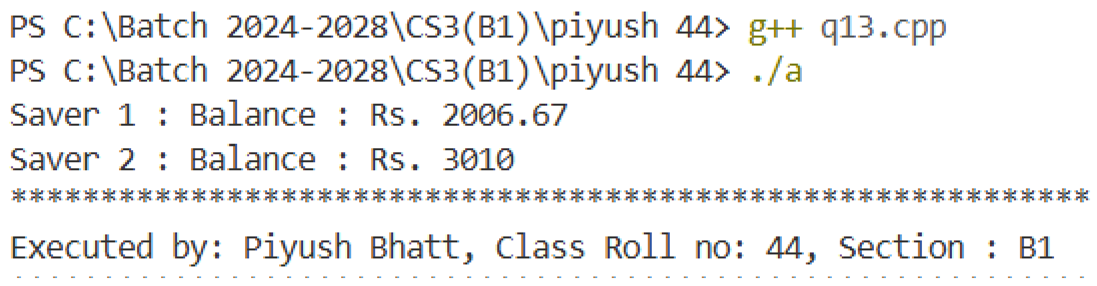
Q13. Create class SavingsAccount. Use a static variable annualInterestRate to store the annual interest rate for all account holders. Each object of the class contains a private instance variable savingsBalance indicating the amount the saver currently has on deposit. Provide method calculateMonthlyInterest() 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 balance;
   static double rate;

public:
   SavingsAccount(double saving)
   {
      balance = saving;
   }
   void calculateMonthlyInterest()
   {
      balance += (balance * rate) / 12;
   }
   static void modify(double newrate)
   {
      rate = newrate;
   }
   void display()
   {
      cout << "Balance : Rs. " << balance << endl;
   }
};
double SavingsAccount ::rate = 0.0;
```

```
int main()
{
   SavingsAccount saver1(2000), saver2(3000);
   SavingsAccount::modify(0.04);
   saver1.calculateMonthlyInterest();
   saver2.calculateMonthlyInterest();
   cout << "Saver 1 : ";
   saver1.display();
   cout << "Saver 2 : ";
   saver2.display();
   cout << "************************************************************" << endl;
   cout << "Executed by: Piyush Bhatt, Class Roll no: 44, Section : B1"  << endl;
   cout << "************************************************************" << endl;
   return 0;
}
```

OUTPUT

Q13.



```
PS C:\Batch 2024-2028\CS3(B1)\piyush 44> g++ q13.cpp
PS C:\Batch 2024-2028\CS3(B1)\piyush 44> ./a
Saver 1 : Balance : Rs. 2006.67
Saver 2 : Balance : Rs. 3010
************************************************************
Executed by: Piyush Bhatt, Class Roll no: 44, Section : B1
------------------------------------------------------------
```

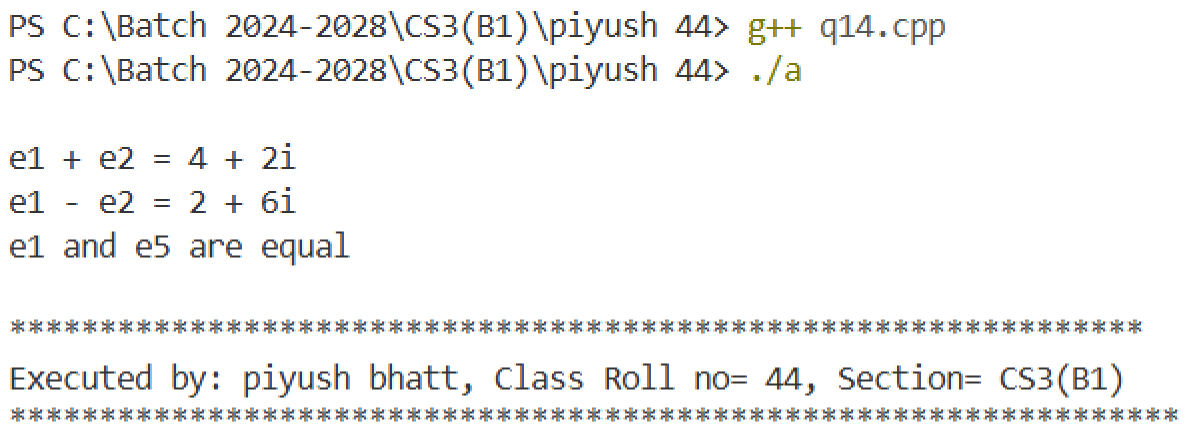
SOURCE CODE

Q14. Create a class Complex having two int type variable named real & img denoting real and imaginary part respectively of a complex number. Overload +, - , == operator to add, to subtract and to compare two complex numbers being denoted by the two complex type objects

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

class Complex {
private:
    int real, img;

public:
    Complex(int r = 0, int i = 0) : real(r), img(i) {}
    Complex operator+(const Complex &c) const {
        Complex result;
        result.real = real + c.real;
        result.img = img + c.img;
        return result;
    }
    Complex operator-(const Complex &c) const {
        Complex result;
        result.real = real - c.real;
        result.img = img - c.img;
        return result;
    }
    bool operator==(const Complex &c) const {
        bool equal = (real == c.real) && (img == c.img);
        return equal;
    }
    void display() const {
        if (img >= 0)
            cout << real << " + " << img << "i" << endl;
        else
            cout << real << " - " << -img << "i" << endl;
    }
};

int main() {
    Complex e1(3, 4);
    Complex e2(1, -2);
    Complex e3, e4;

    e3 = e1 + e2;
    cout << "\ne1 + e2 = ";
    e3.display();

    e4 = e1 - e2;
    cout << "e1 - e2 = ";
    e4.display();

    Complex e5(3, 4);
    if (e1 == e5) {
        cout << "e1 and e5 are equal" << endl;
    } else {
        cout << "e1 and e5 are not equal" << endl;
    }
    cout<<"\n***************************************************************\n";
    cout<<"Executed by: piyush bhatt, Class Roll no= 44, Section= CS3(B1)"<<endl;
    cout<<"******************************************************************\n";
    return 0;
}
```

OUTPUT

Q14.



```
PS C:\Batch 2024-2028\CS3(B1)\piyush 44> g++ q14.cpp
PS C:\Batch 2024-2028\CS3(B1)\piyush 44> ./a

e1 + e2 = 4 + 2i
e1 - e2 = 2 + 6i
e1 and e5 are equal

***************************************************************
Executed by: piyush bhatt, Class Roll no= 44, Section= CS3(B1)
*****************************************************************
```

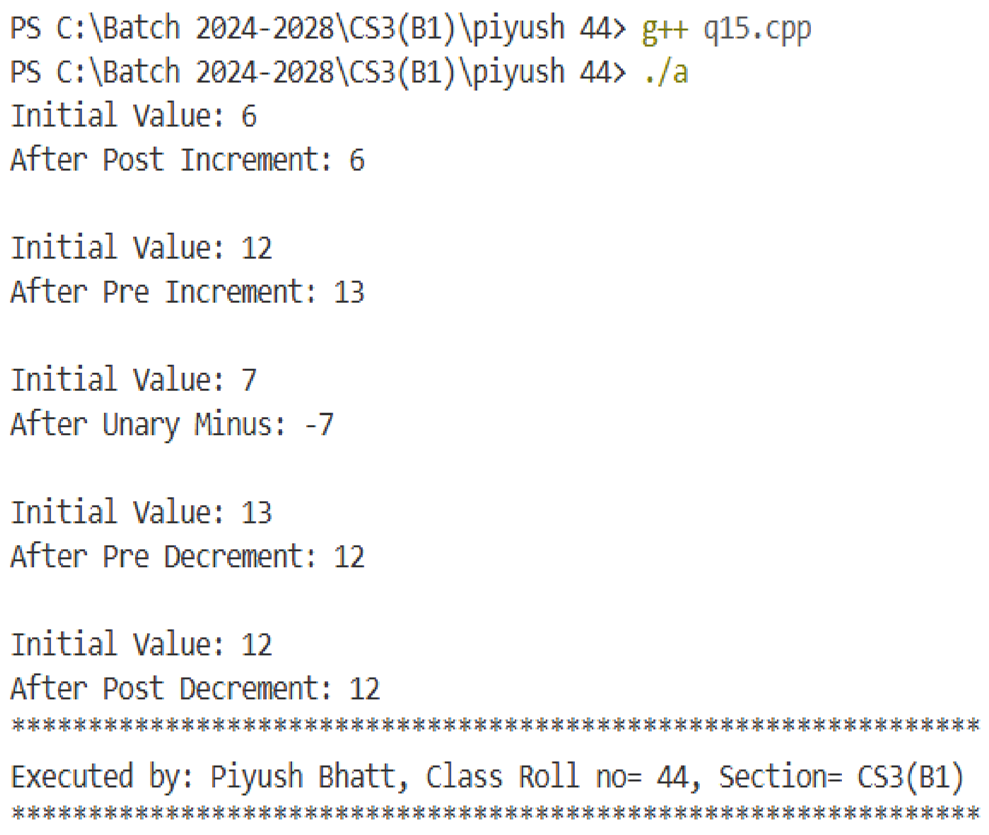
SOURCE CODE

Q15. Using the concept of operator overloading. Implement a program to overload the following:
a. Unary –
b. Unary ++ preincrement, postincrement
c. Unary -- predecrement, postdecrement

```
#include  <iostream>
using namespace std;
class Sample {
   int value;
public:
   Sample(int v = 0) : value(v) {}
   Sample operator++() {
      ++value;
      return *this;
   }
   Sample operator++(int) {
      Sample temp = *this;
      value++;
      return temp;
   }
   Sample operator--() {
      --value;
      return *this;
   }
   Sample operator--(int) {
      Sample temp = *this;
      value--;
      return temp;
   }
   Sample operator-() const {
      Sample temp;
      temp.value = -value;
      return temp;
   }
   void display() const {
      cout << value << endl;
   }
};

int main() {
    Sample c1(6), c2, c3(12), c4, c5, c6, c7;
    cout << "Initial Value: ";
    c1.display();
    c2 = c1++;
    cout << "After Post Increment: ";
    c2.display();
    cout << endl;
    cout << "Initial Value: ";
    c3.display();
    c4 = ++c3;
    cout << "After Pre Increment: ";
    c4.display();
    cout << endl;
    cout << "Initial Value: ";
    c1.display();
    c5 = -c1;
    cout << "After Unary Minus: ";
    c5.display();
    cout << endl;
    cout << "Initial Value: ";
    c3.display();
    c6 = --c3;
    cout << "After Pre Decrement: ";
    c6.display();
    cout << endl;
    cout << "Initial Value: ";
    c6.display();
    c7 = c6--;
    cout << "After Post Decrement: ";
    c7.display();
    cout << "**************************************************************\n";
    cout << “Executed by: piyush bhatt, Class Roll no= 44, Section= CS3(B1)\n";
    cout << "**************************************************************\n";
    return 0;
}
```

OUTPUT

Q15.



```
PS C:\Batch 2024-2028\CS3(B1)\piyush 44> g++ q15.cpp
PS C:\Batch 2024-2028\CS3(B1)\piyush 44> ./a
Initial Value: 6
After Post Increment: 6

Initial Value: 12
After Pre Increment: 13

Initial Value: 7
After Unary Minus: -7

Initial Value: 13
After Pre Decrement: 12

Initial Value: 12
After Post Decrement: 12
***************************************************************
Executed by: Piyush Bhatt, Class Roll no= 44, Section= CS3(B1)
***************************************************************
```

SOURCE CODE

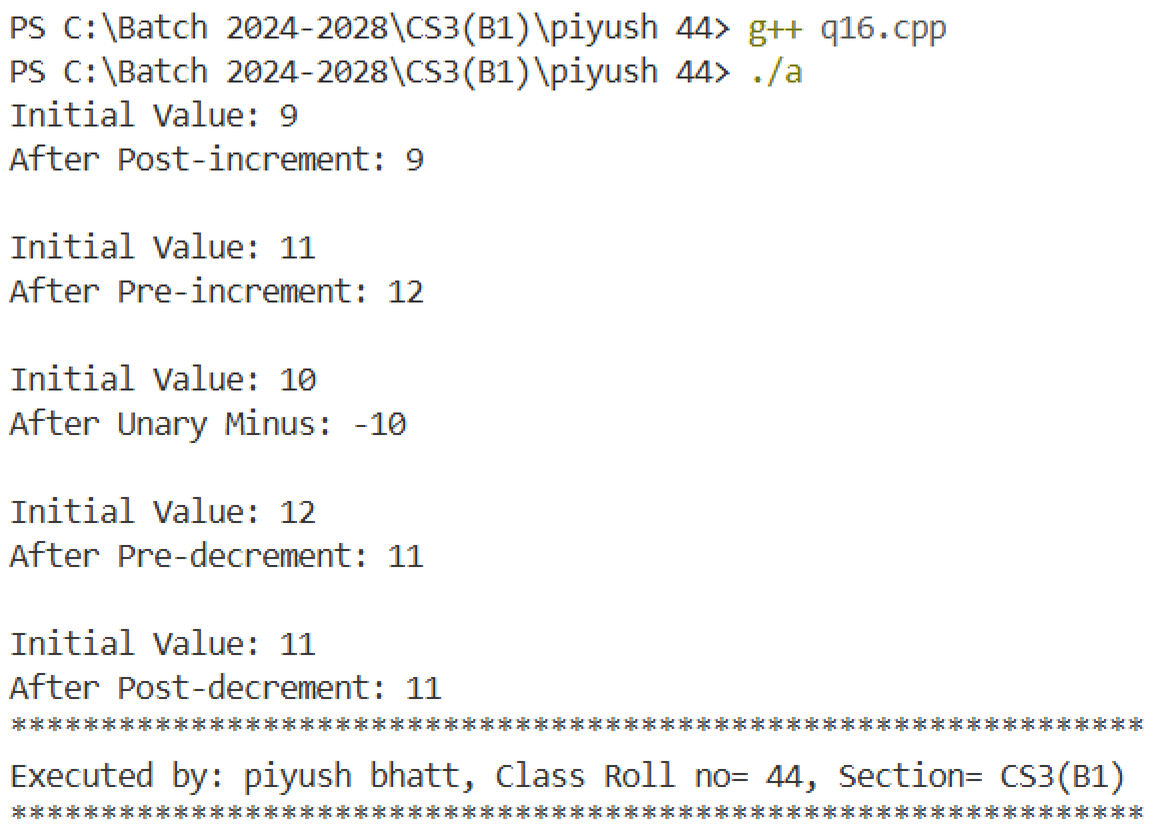
Q16. Using the concept of operator overloading. Implement a program to overload the following:
With the help of friend function
a. Unary –
b. Unary ++ preincrement, postincrement
c. Unary -- predecrement, postdecrement

```
#include  <iostream>
using namespace std;
class Numbers {
   int value;
public:
   Numbers(int v = 0) : value(v) {}
   friend Numbers operator++(Numbers &s);
   friend Numbers operator++(Numbers &s, int);
   friend Numbers operator--(Numbers &s);
   friend Numbers operator--(Numbers &s, int);
   friend Numbers operator-(const Numbers &s);
   void display() const {
      cout << value << endl;
   }
};
Numbers operator++(Numbers &s) {
   ++s.value;
   return s;
}
Numbers operator++(Numbers &s, int) {
   Numbers temp = s;
   s.value++;
   return temp;
}
Numbers operator--(Numbers &s) {
   --s.value;
   return s;
}
Numbers operator--(Numbers &s, int) {
   Numbers temp = s;
   s.value--;
   return temp;
}
Numbers operator-(const Numbers &s) {
   Numbers temp;
```

```
    temp.value = -s.value;
    return temp;
}

int main() {
    Numbers c1(9), c2, c3(11), c4, c5, c6, c7;
    cout << "Initial Value: ";
    c1.display();
    c2 = c1++;
    cout << "After Post-increment: ";
    c2.display();
    cout << endl;
    cout << "Initial Value: ";
    c3.display();
    c4 = ++c3;
    cout << "After Pre-increment: ";
    c4.display();
    cout << endl;
    cout << "Initial Value: ";
    c1.display();
    c5 = -c1;
    cout << "After Unary Minus: ";
    c5.display();
    cout << endl;
    cout << "Initial Value: ";
    c3.display();
    c6 = --c3;
    cout << "After Pre-decrement: ";
    c6.display();
    cout << endl;
    cout << "Initial Value: ";
    c6.display();
    c7 = c6--;
    cout << "After Post-decrement: ";
    c7.display();
    cout << "****************************************************************\n";
    cout << “Executed by: piyush bhatt, Class Roll no= 44, Section= CS3(B1)\n";
    cout << "****************************************************************\n";
    return 0;
}
```

OUTPUT

Q16.



```
PS C:\Batch 2024-2028\CS3(B1)\piyush 44> g++ q16.cpp
PS C:\Batch 2024-2028\CS3(B1)\piyush 44> ./a
Initial Value: 9
After Post-increment: 9

Initial Value: 11
After Pre-increment: 12

Initial Value: 10
After Unary Minus: -10

Initial Value: 12
After Pre-decrement: 11

Initial Value: 11
After Post-decrement: 11
****************************************************************
Executed by: piyush bhatt, Class Roll no= 44, Section= CS3(B1)
****************************************************************
```

SOURCE CODE

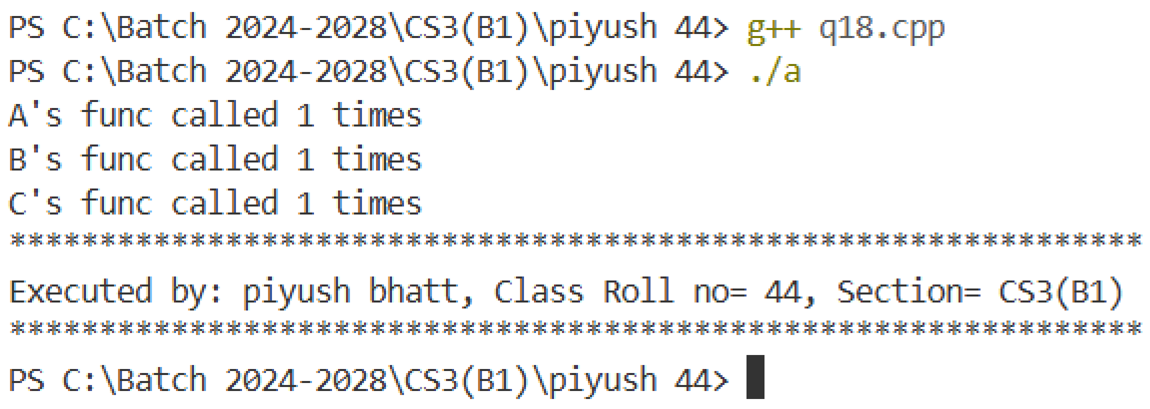
Q18. You are given three classes A, B and C. All three classes implement their own version of func. In class A, func multiplies the value passed as a parameter by 2. In class B, func multiplies the value passed as a parameter by 3. In class C, func multiplies the value passed as a parameter by 5.You are given class D such that You need to modify the class D and implement the function update_val which sets D's val to new_val by manipulating the value by only calling the func defined in classes A, B and C.It is guaranteed that new_val has only 2, 3 and 5 as its prime factors. Implement class D's function update_val. This function should update D's val only by calling A, B and C's func. Sample Input new_val = 30 Sample Output A's func called 1 times B's func called 1 times C's func called 1 times

```
#include <iostream>
 using namespace std;
 class A {
 protected:
 int callA;
 public:
 A() : callA(0) {}
 void func(int &val) {
 val *= 2;
 callA++;
 }
 int getA() { return callA; }
 };
 class B {
 protected:
 int callB;
 public:
 B() : callB(0) {}
 void func(int &val) {
 val *= 3;
 callB++;
 }
 int getB() { return callB; }
 };
 class C {
 protected:
 int callC;
 public:
 C() : callC(0) {}
 void func(int &val) {
```

```
val *= 5;
callC++;
}
int getC() { return callC; }
};
class D : public A, public B, public C {
int val;
public:
D() : val(1) {}
void update_val(int new_val) {
while (val < new_val) {
if (new_val % (val * 2) == 0) {
A::func(val);
} else if (new_val % (val * 3) == 0) {
B::func(val);
} else if (new_val % (val * 5) == 0) {
C::func(val);
}
}
}
void check(int new_val) {
update_val(new_val);
cout << "A's func called " << getA() << " times" << endl;
cout << "B's func called " << getB() << " times" << endl;
cout << "C's func called " << getC() << " times" << endl;
}
};
int main() {
int new_val = 30;
D d;
d.check(new_val);

cout << "****************************************************************\n";
cout << "Executed by: piyush bhatt, Class Roll no= 44, Section= CS3(B1)\n";
cout << "****************************************************************\n";

return 0;
}
```

OUTPUT

Q18.



```
PS C:\Batch 2024-2028\CS3(B1)\piyush 44> g++ q18.cpp
PS C:\Batch 2024-2028\CS3(B1)\piyush 44> ./a
A's func called 1 times
B's func called 1 times
C's func called 1 times
*****************************************************************
Executed by: piyush bhatt, Class Roll no= 44, Section= CS3(B1)
*****************************************************************
PS C:\Batch 2024-2028\CS3(B1)\piyush 44> 
```

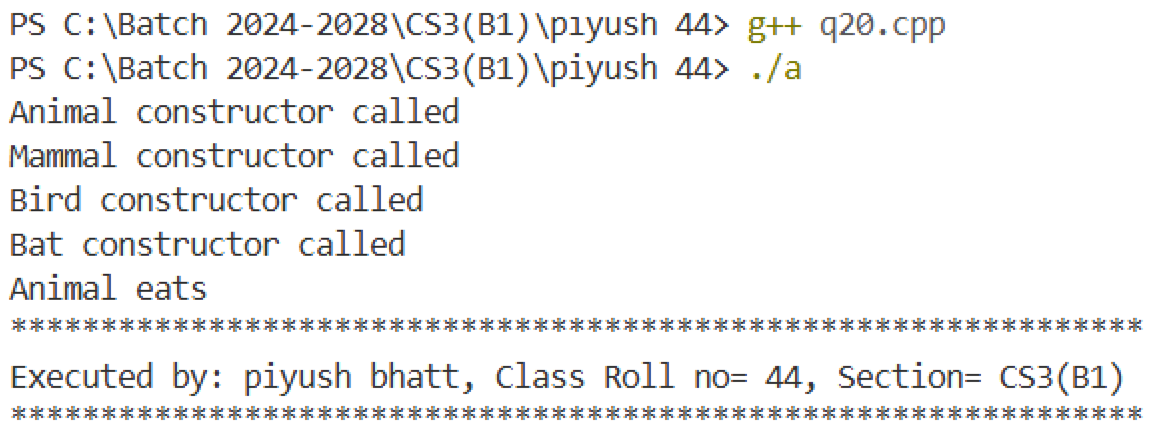
SOURCE CODE

Q20. Create a class called Student that contains the data members like age, name, enroll_no, marks. Create another class called Faculty that contains data members like facultyName, facultyCode, salary,deptt, age, experience, gender. Create the function display() in both the classes to display the respective information. The derived Class Person demonstrates multiple inheritance. The program should be able to call both the base classes and displays their information. Remove the ambiguity (When we have exactly same variables or same methods in both the base classes, which one will becalled?) by proper mechanism.

```
#include <iostream>
using namespace std;
class Animal {
public:
    Animal() { cout << "Animal constructor called\n"; }
    void eat() { cout << "Animal eats\n"; }
};
class Mammal : virtual public Animal {
public:
    Mammal() { cout << "Mammal constructor called\n"; }
};
class Bird : virtual public Animal {
public:
    Bird() { cout << "Bird constructor called\n"; }
};
class Bat : public Mammal, public Bird {
public:
    Bat() { cout << "Bat constructor called\n"; }
};
int main() {
    Bat b;
    b.eat();

    cout << "**************************************************************\n";
    cout << "Executed by: piyush bhatt, Class Roll no= 44, Section= CS3(B1)\n";
    cout << "**************************************************************\n";
    return 0;
}
```

OUTPUT

Q20.



```
PS C:\Batch 2024-2028\CS3(B1)\piyush 44> g++ q20.cpp
PS C:\Batch 2024-2028\CS3(B1)\piyush 44> ./a
Animal constructor called
Mammal constructor called
Bird constructor called
Bat constructor called
Animal eats
****************************************************************
Executed by: piyush bhatt, Class Roll no= 44, Section= CS3(B1)
****************************************************************
```

SOURCE CODE

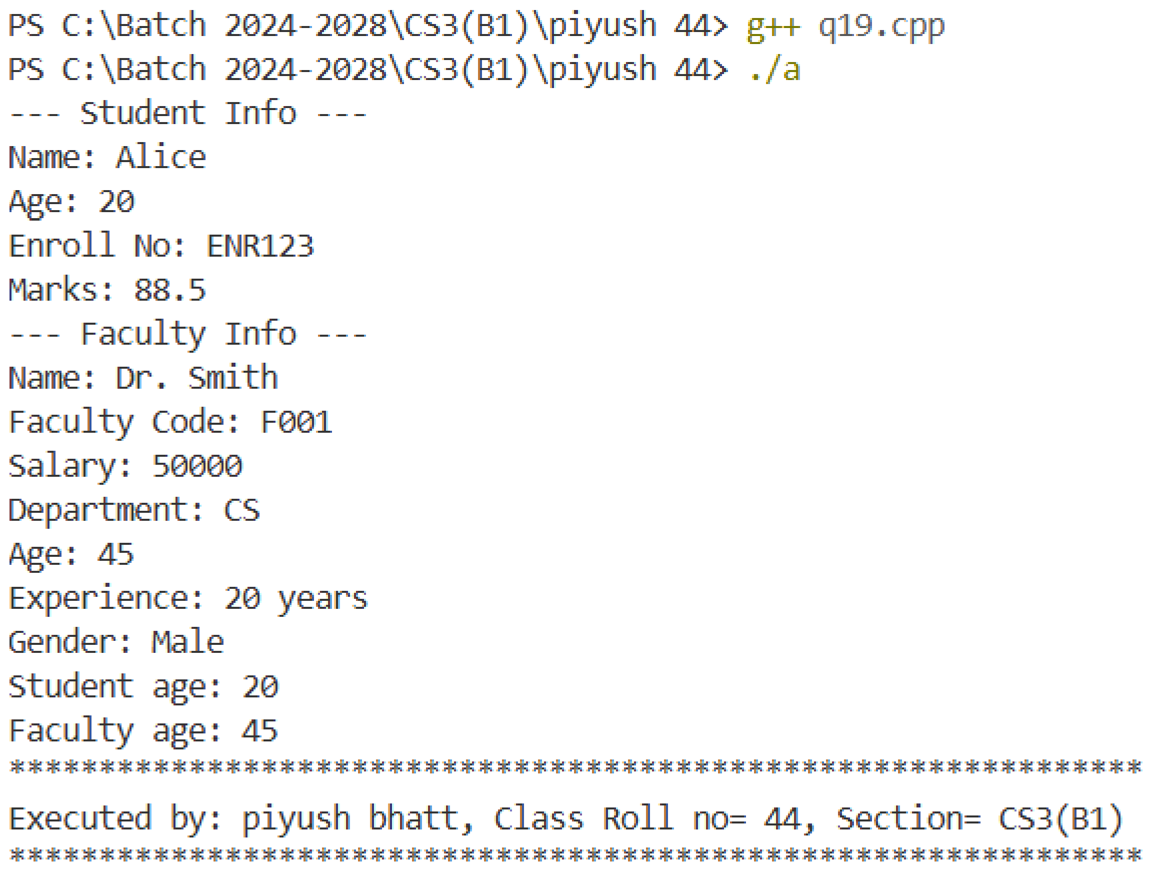
Q19. Create a class called Student that contains the data members like age, name, enroll_no, marks. Create another class called Faculty that contains data members like facultyName, facultyCode, salary,deptt, age, experience, gender. Create the function display() in both the classes to display the respective information. The derived Class Person demonstrates multiple inheritance. The program should be able to call both the base classes and displays their information. Remove the ambiguity (When we have exactly same variables or same methods in both the base classes, which one will becalled?) by proper mechanism.

```
#include <iostream>
#include <string>
using namespace std;
class Student {
protected:
   int age;
   string name;
   string enroll_no;
   float marks;
public:
   Student(int age, string name, string enroll_no, float marks)
      : age(age), name(name), enroll_no(enroll_no), marks(marks) {}
   void display() {
      cout << "--- Student Info ---" << endl;
      cout << "Name: " << name << endl;
      cout << "Age: " << age << endl;
      cout << "Enroll No: " << enroll_no << endl;
      cout << "Marks: " << marks << endl;
   }
};
class Faculty {
protected:
   string facultyName;
   string facultyCode;
   float salary;
   string deptt;
   int age;
   int experience;
   string gender;
public:
   Faculty(string facultyName, string facultyCode, float salary, string deptt, int age, int
experience, string gender)
  : facultyName(facultyName), facultyCode(facultyCode), salary(salary), deptt(deptt),
age(age), experience(experience), gender(gender) {}
   void display() {
      cout << "--- Faculty Info ---" << endl;
      cout << "Name: " << facultyName << endl;
      cout << "Faculty Code: " << facultyCode << endl;
      cout << "Salary: " << salary << endl;
```

```
        cout << "Department: " << deptt << endl;
        cout << "Age: " << age << endl;
        cout << "Experience: " << experience << " years" << endl;
        cout << "Gender: " << gender << endl;
    }
};
class Person : public Student, public Faculty {
public:
    Person(int s_age, string s_name, string enroll_no, float marks,
           string f_name, string f_code, float salary, string deptt, int f_age, int exp, string gender)
        : Student(s_age, s_name, enroll_no, marks),
          Faculty(f_name, f_code, salary, deptt, f_age, exp, gender) {}
    void display() {
        Student::display();
        Faculty::display();
        cout << "Student age: " << Student::age << endl;
        cout << "Faculty age: " << Faculty::age << endl;
    }
};
int main() {
    Person p(20, "Alice", "ENR123", 88.5, "Dr. Smith", "F001", 50000, "CS", 45, 20, "Male");
    p.display();
     cout << "************************************************************\n";
    cout << "Executed by: piyush bhatt, Class Roll no= 44, Section= CS3(B1)\n";
    cout << "************************************************************\n";
    return 0;
}
```

OUTPUT

Q19.



```
PS C:\Batch 2024-2028\CS3(B1)\piyush 44> g++ q19.cpp
PS C:\Batch 2024-2028\CS3(B1)\piyush 44> ./a
--- Student Info ---
Name: Alice
Age: 20
Enroll No: ENR123
Marks: 88.5
--- Faculty Info ---
Name: Dr. Smith
Faculty Code: F001
Salary: 50000
Department: CS
Age: 45
Experience: 20 years
Gender: Male
Student age: 20
Faculty age: 45
***************************************************************
Executed by: piyush bhatt, Class Roll no= 44, Section= CS3(B1)
***************************************************************
```

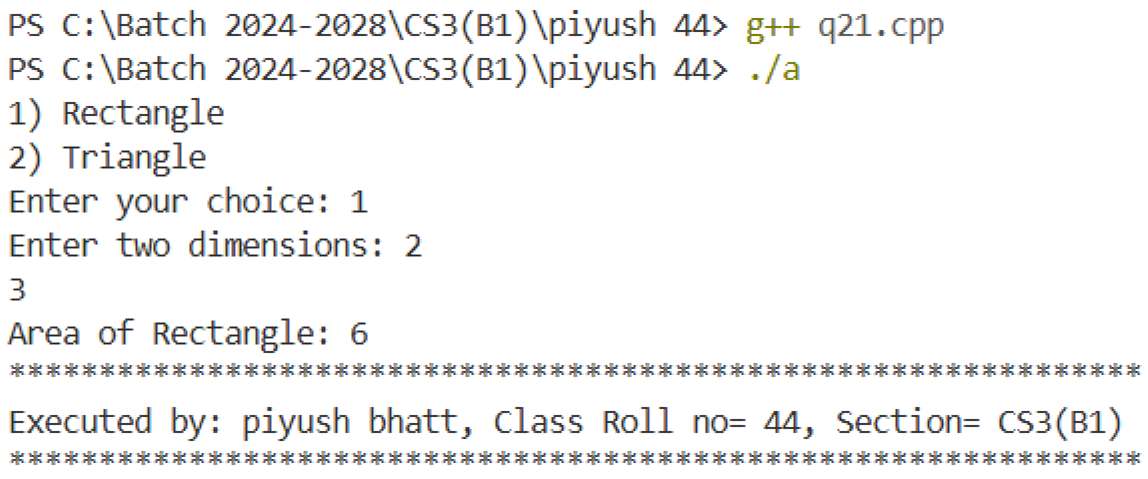
SOURCE CODE

Q21. Create a base class called shape. Use this class to store two double type values that could be used to compute the area of figures. Derive two specific classes called triangle and rectangle from base shape. Add to the base class, a member function get_data() to initialize base class data members and another member function display_area() to compute and display the area of figures. Make display_area() as a virtual function and redefine this function in the derived class to suit their requirements. Using these three classes, design a program that will accept dimensions of a triangle or a rectangle interactively and display the area. Remember the two values given as input will be treated as lengths of two sides in the case of rectangles and as base and height in the case of triangle and used as follows: Area of rectangle = x * y Area of triangle = ½ *x*y.

```
#include <iostream>
#include <cmath>
using namespace std;
const float PI = 3.14;
class CAL_AREA{
protected:
    float r, h;
public:
    virtual void getdata(){
        cout << "Enter radius: ";
        cin >> r;
    }
    virtual void display_volume() = 0;
};
class cone : public CAL_AREA{
public:
    void getdata(){
        cout << "Enter radius and height of cone: ";
        cin >> r >> h;
    }
    void display_volume(){
        float volume = (1.0f/3.0f) * PI * r * r * h;
        cout << "Volume of Cone: " << volume << endl;
    }
};
class hemisphere : public CAL_AREA{
public:
    void getdata(){
          cout << "Enter radius of hemisphere: ";
        cin >> r;
    }
    void display_volume(){
        float volume = (2.0f/3.0f) * PI * r * r * r;
        cout << "Volume of Hemisphere: " << volume << endl;
    }
};
class cylinder : public CAL_AREA {
public:
    void getdata(){
        cout << "Enter radius and height of cylinder: ";
        cin >> r >> h;
    }
    void display_volume(){
        float volume = PI * r * r * h;
        cout << "Volume of Cylinder: " << volume << endl;
    }
};
int main(){
    CAL_AREA* ptr;
    int choice;
    cout << "1) Cone\n2) Hemisphere\n3) Cylinder\nEnter your choice: ";
    cin >> choice;
    if(choice == 1) {
        cone c;
        ptr = &c;
    } else if(choice == 2) {
        hemisphere h;
        ptr = &h;
    } else if(choice == 3) {
        cylinder cy;
        ptr = &cy;
    } else {
        cout << "Invalid choice!" << endl;
        return 0;
    }
    ptr->getdata();
    ptr->display_volume();
    return 0;
}
```

OUTPUT

Q21.



```
PS C:\Batch 2024-2028\CS3(B1)\piyush 44> g++ q21.cpp
PS C:\Batch 2024-2028\CS3(B1)\piyush 44> ./a
1) Rectangle
2) Triangle
Enter your choice: 1
Enter two dimensions: 2
3
Area of Rectangle: 6
****************************************************************
Executed by: piyush bhatt, Class Roll no= 44, Section= CS3(B1)
****************************************************************
```

SOURCE CODE

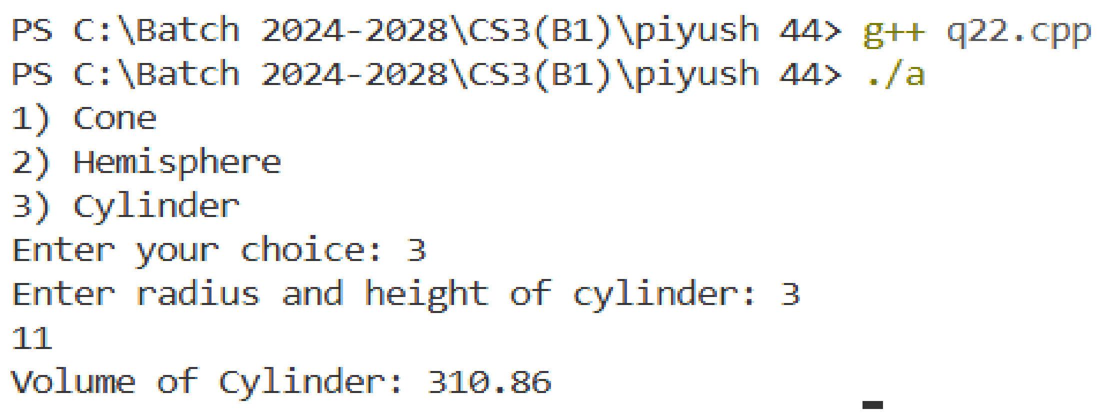
Q22. Create a base class called CAL_AREA(Abstract). Use this class to store float type values that could be used to compute the volume of figures. Derive two specific classes called cone, hemisphere and cylinder from the base CAL_AREA. Add to the base class, a member function getdata ( ) to initialize base class data members and another member function display volume( ) to compute and display the volume of figures. Make display volume ( ) as a pure virtual function and redefine this function in the derived classes to suit their requirements. Using these three classes, design a program that will accept dimensions of a cone, cylinder and hemisphere interactively and display the volumes. Remember values given as input will be and used as follows: Volume of cone = (1/3)πr2h Volume of hemisphere = (2/3)πr3 Volume of cylinder = πr2h

```
#include <iostream>
#include <cmath>
using namespace std;
const float PI = 3.14;
class CAL_AREA{
protected:
   float r, h;
public:
   virtual void getdata(){
      cout << "Enter radius: ";
      cin >> r;
   }
   virtual void display_volume() = 0;
};
class cone : public CAL_AREA{
public:
   void getdata(){
      cout << "Enter radius and height of cone: ";
      cin >> r >> h;
   }
   void display_volume(){
      float volume = (1.0f/3.0f) * PI * r * r * h;
      cout << "Volume of Cone: " << volume << endl;
   }
};
class hemisphere : public CAL_AREA{
public:
   void getdata(){
       cout << "Enter radius of hemisphere: ";
      cin >> r;
   }
   void display_volume(){
      float volume = (2.0f/3.0f) * PI * r * r * r;
```

```
  cout << "Volume of Hemisphere: " << volume << endl;
    }
};
class cylinder : public CAL_AREA {
public:
    void getdata(){
        cout << "Enter radius and height of cylinder: ";
        cin >> r >> h;
    }
    void display_volume(){
        float volume = PI * r * r * h;
        cout << "Volume of Cylinder: " << volume << endl;
    }
};
int main(){
    CAL_AREA* ptr;
    int choice;
    cout << "1) Cone\n2) Hemisphere\n3) Cylinder\nEnter your choice: ";
    cin >> choice;
    if(choice == 1) {
        cone c;
        ptr = &c;
    } else if(choice == 2) {
        hemisphere h;
        ptr = &h;
    } else if(choice == 3) {
        cylinder cy;
        ptr = &cy;
    } else {
        cout << "Invalid choice!" << endl;
        return 0;
    }
    ptr->getdata();
    ptr->display_volume();
    return 0;
}
```

OUTPUT

Q22.



```
PS C:\Batch 2024-2028\CS3(B1)\piyush 44> g++ q22.cpp
PS C:\Batch 2024-2028\CS3(B1)\piyush 44> ./a
1) Cone
2) Hemisphere
3) Cylinder
Enter your choice: 3
Enter radius and height of cylinder: 3
11
Volume of Cylinder: 310.86
```

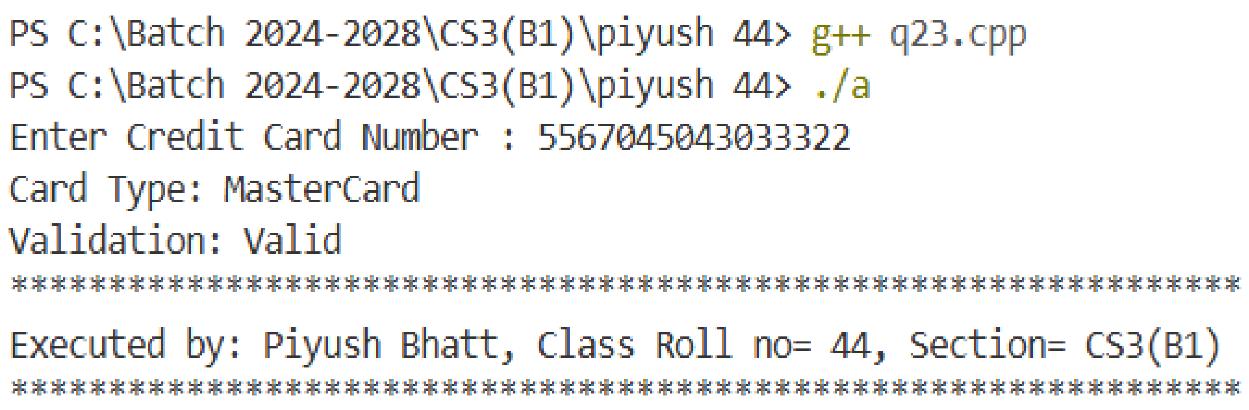
SOURCE CODE

Q23. Solve the following problem by using runtime polymorphism: Following tables outlines the major credit cards you might want to validate, along with their allowed prefixes and lengths.

```
#include<iostream>
#include<string>
using namespace std;
class CreditCard
{
   public:
   string cardnumber;
   virtual bool validate(){
      return false;
   }
   virtual string getCardType(){
      return "UNKNOWN";
   }
   virtual ~CreditCard(){}
};
class MasterCard : public CreditCard
{
   public:
   bool validate(){
      if(cardnumber.length()==16){
         int prefix = stoi(cardnumber.substr(0,2));
         return (prefix>=51 && prefix <=55);
      }return false;
   }
   string getCardType(){
      return "MasterCard";
   }
};
class Visa : public CreditCard
{
   public:
   bool validate(){
      if((cardnumber.length()==16|| cardnumber.length()==13)&& cardnumber[0]=='4'){
         return true;
      }
      return false;
   }
    string getCardType(){
         return "VisaCard";
   }
};
class AmericanExpress : public CreditCard
{
   public:
   bool validate(){
        if(cardnumber.length()==15){
            string prefix = cardnumber.substr(0,2);
            return (prefix == "34" || prefix == "37");
        }
        return false;
    }
     string getCardType(){
        return "AmericanExpress";
    }
};
int main(){
    CreditCard * card = nullptr;
    string input;
    cout << "Enter Credit Card Number : ";
    cin >> input;
    CreditCard* cards[3] = { new MasterCard(), new Visa(), new AmericanExpress() };
    bool found = false;
    for(int i=0; i<3; ++i) {
        cards[i]->cardnumber = input;
        if(cards[i]->validate()) {
            card = cards[i];
            found = true;
            break;
        }
    }
    if(found) {
        cout << "Card Type: " << card->getCardType() << endl;
        cout << "Validation: Valid" << endl;
    } else {
        cout << "Invalid or unknown Card type" << endl;
    }
    for(int i=0; i<3; ++i) {
        if(cards[i] != card) delete cards[i];
    }
    delete card;

    cout << "***************************************************************\n";
    cout << "Executed by: Piyush Bhatt, Class Roll no= 44, Section= CS3(B1)\n";
    cout << "***************************************************************\n";

    return 0;
}
```

OUTPUT

Q23.



```
PS C:\Batch 2024-2028\CS3(B1)\piyush 44> g++ q23.cpp
PS C:\Batch 2024-2028\CS3(B1)\piyush 44> ./a
Enter Credit Card Number : 5567045043033322
Card Type: MasterCard
Validation: Valid
*****************************************************************
Executed by: Piyush Bhatt, Class Roll no= 44, Section= CS3(B1)
*****************************************************************
```