OOP WITH C++ LAB TERM WORK

CODE:PCS 307

Q1. Write a *C++ program to create a function* void arraySort(int A[], int p, int B[], int q) .Given two sorted arrays A and B of size p and q, define function
arraySort() to merge elements of A with B by maintaining the sorted order i.e. fill A with first p smallest elements and fill B with remaining elements.

Sample Input :
int[] A = { 1, 5, 6, 7, 8, 10 }
int[] B = { 2, 4, 9 }

Output:
Sorted Arrays:
A: [1, 2, 4, 5, 6, 7]
B: [8, 9, 10]

Program:

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

void arraySort(int A[], int B[], int p, int q,int C[])
{
    int i = 0, j = 0, k = 0;

    while (i<p && j <q)
    {
        if (A[i] < B[j])
            C[k++] = A[i++];
        else
            C[k++] = B[j++];
    }

    while (i < p)
        C[k++] = A[i++];

    while (j < q)
        C[k++] = B[j++];
}
```

```
int main()
{
    int p,q;
    cout<<"Enter the Size of array A = "<<endl;
    cin>>p;
    int A[p];
    cout<<"Enter the array A element : "<<endl;
    for(int i=0;i<p;i++)
    {
        cin>>A[i];
    }

    cout<<"Enter the Size of array B = "<<endl;
    cin>>q;
    int B[q];
    cout<<"Enter the array A element : "<<endl;
    for(int j=0;j<q;j++)
    {
        cin>>B[j];
    }

    int C[p+q];
    arraySort(A, B, p, q, C);

   int k;
   cout<<"A : ";
   for(k=0;k<p;k++)
   {
      cout<<C[k]<<" ";
   }
   cout<<endl;
   cout<<"B : ";
   for(k=p;k<p+q;k++)
   {
      cout<<C[k]<<" ";
   }

   return 0;
}
```

Output:

```
PS D:\practice c++\term work c++> cd "d:\practice c++\term work c++\" ; if ($?) { g++ q1.cpp -o q1 } ;
if ($?) { .\q1 }
Enter the Size of array A =
6
Enter the array A element :
```

```
1
5
6
7
8
10
Enter the Size of array B =
3
Enter the array A element :
2
4
9
A : 1 2 4 5 6 7
B : 8 9 10
PS D:\practice c++\term work c++>
```

Q2. Write a C++ program to input any string and arrange all characters in following order.
Digit + Uppercase + Lowercase + Special Characters

Sample Example:
Input: GrAphic567E#@RA
Output: 567GAERArphic#@

Program:

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

int main()
{
   string str, d, u, l, sc, total;

   cout<<"Input : ";
   getline(cin,str);

   for(int i=0;str[i]!='\0';i++)
   {
      if(str.at(i)>='0' && str.at(i)<='9')
      {
         d = d+str.at(i);
      }
      else if(str.at(i)>='A' &&
str.at(i)<='Z')
      {
         u = u+str.at(i);
      }
      else if(str.at(i)>='a' && str.at(i)<='z')
      {
         l = l+str.at(i);
      }
      else
```

```
            {
               sc = sc+str.at(i);
            }
        }

        total = d + u + l + sc;
        cout<<"Output : "<<total;
    }
```

Output:

```
PS D:\practice c++\term work c++> cd
"d:\practice c++\term work c++\" ; if
($?) { g++ q2.cpp -o q2 } ; if ($?) { .\q2
}
Input : GrAphic567E#@RA
Output : 567GAERArphic#@
PS D:\practice c++\term work c++>
```

Q3.Define a class employee having the following description:

Data Members	**Description**
Pan	To store personal account number
Name	To store name
Taxincome	To store annual taxable income
Tax	To store tax that is calculated

Member Functions	**Description**

InputInfo()	Store the pan number,name,taxableincome
TaxCalc()	Calculate tax for an employee
DisplayInfo()	Output details of an employee

Write a C++ program to compute the tax according to the given conditions and display the output.

Total Annual Taxable Income	**Tax Rate**
Upto 2,50,000	No tax
From 2,50,000 to 3,00,000	10 % of the income exceeding 2,50,000
From 3,00,000 to 4,00,000	Rs. 5000+20 % of the income exceeding 3,00,000
Above 4,00,000	Rs 25000 + 30 % of the income exceeding 4,00,000

Program:

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

class employee
{
   private:
   int pan;
   float taxincome;
   float tax;
   string name;

   public:

   void inputinfo();

   void calc();

   void displayinfo();

};

void employee :: inputinfo()
{
   cout<<"Enter name : ";
```

```
    getline(cin,name);
    cout<<"Enter Pan number : ";
    cin>>pan;
    cout<<"Enter taxable income : ";
    cin>>taxincome;
}

void employee :: calc()
{
    if (taxincome <= 250000)
       tax = 0;
    else if (taxincome <= 300000)
        tax = (taxincome - 250000) * 0.1;
    else if (taxincome <= 400000)
        tax = 5000 + ((taxincome - 300000) * 0.2);
    else
        tax = 25000 + ((taxincome - 400000) * 0.3);
}

void employee :: displayinfo()
{
    cout<<"Pan number : "<<pan<<endl<<"Name : "<<name<<endl<<"Taxable income : 
"<<taxincome<<endl<<"tax : "<<tax;
}

int main()
{
    employee e1;
    e1.inputinfo();
    e1.calc();
    e1.displayinfo();

    return 0;
}
```

Output:

```
PS D:\practice c++\term work c++> cd "d:\practice c++\term work c++\" ; if ($?) {
g++ q3.cpp -o q3 } ; if ($?) { .\q3 }
Enter name : Abhishek Singh
Enter Pan number : 4852
Enter taxable income : 600000
Pan number : 4852
Name : Abhishek Singh
Taxable income : 600000
tax : 85000
PS D:\practice c++\term work c++>
```

Q4.W.A.P 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(ASSUME)

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

Program:

```
#include <iostream>
#include <conio.h>
#include <string.h>
using namespace std;
class bank
{
   char name[20];
   int ano;
   char atype[20];
   float bal;

public:
   void get(int no, char *n, char *t, float b)
   {
      strcpy(name, n);
      ano = no;
      strcpy(atype, t);
      bal = b;
   }
   float deposit()
   {
      float amt;
      cout << "\nEnter amount: ";
      cin >> amt;
      bal = bal + amt;
      return bal;
   }
   float withdrw()
   {
      float amt;
      cout << "\nHow many Rupees withdraw: ";
      cin >> amt;
      bal = bal - amt;
      return bal;
   }
   void disp()
   {
        cout << "\n\nAccount number: " << ano;
        cout << "\n\nName: " << name;
        cout << "\n\nAccount type: " << atype;
        cout << "\n\nDeposit Amount: " << deposit();
        cout << "\n\nAfter Withdraw Amount balnace: " << withdrw();
    }
};
int main()
{
    int n;
    char nm[20], t[20];
    float a;
    bank bk;
    cout << "\nEnter Account no.: ";
    cin >> n;
    cout << "\nEnter Name: ";
    cin >> nm;
    cout << "\nEnter account type: ";
    cin >> t;
    cout << "\nEnter balance amount: ";
    cin >> a;
    bk.get(n, nm, t, a);
    bk.disp();
    getch();
}
```

Output:

```
 Enter Details:
-----------------------
 Accout No. 4587962

 Name : Abhishek

 Account Type : Savings

 Balance : 34000

 Enter Deposit Amount = 3000

 Enter Withdraw Amount = 25000

 ----------------------
 Accout No. : 4587962
 Name : Abhishek
 Account Type : Savings
 Balance : 12000
```

Q5.Imagine a tollbooth with a class called TollBooth. The two data itemsare 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.

Program:

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

class toolbooth
{
   unsigned int car;
   double am;
   public:
   toolbooth()
   {
      car = 0;
      am = 0;
   }
   void payingCar()
   {
      car++;
      am = am+0.5;
   }
   void nonPayCar()
   {
      car++;
   }
   void display()
   {
      cout<<"Total car : "<<car<<endl;
      cout<<"Total amount : "<<am<<endl;
   }
};
int main()
{
   toolbooth tb;
   int ch;
   do
   {
      cout<<"Enter Your choice : "<<endl;

      cout<<"1.Paying car"<<endl<<"2.Non Paying car"<<endl<<"ESC.Total"<<endl;
      cin>>ch;

      switch(ch)
      {
          case 1:
          {
              tb.payingCar();
              tb.display();
              break;
          }
          case 2:
          {
              tb.nonPayCar();
              tb.display();
              break;
          }
          case 3:
          {
              tb.display();
              exit(0);
              break;
          }
      }
   }while(ch!=0);

   return 0;
}
```

Output:

```
PS D:\practice c++\term work c++> cd "d:\practice c++\term work c++\" ; if ($?) { g++ q5.cpp -
o q5 } ; if ($?) { .\q5 }
Enter Your choice :
1.Paying car
2.Non Paying car
ESC.Total
1
Total car : 1
Total amount : 0.5
Enter Your choice :
1.Paying car
2.Non Paying car
ESC.Total
2
```

```
Total car : 2
Total amount : 0.5
Enter Your choice :
1.Paying car
2.Non Paying car
ESC.Total
1
Total car : 3
Total amount : 1
Enter Your choice :
1.Paying car
2.Non Paying car
ESC.Total
1
Total car : 4
Total amount : 1.5
Enter Your choice :
1.Paying car
2.Non Paying car
ESC.Total
2
Total car : 5
Total amount : 1.5
Enter Your choice :
1.Paying car
2.Non Paying car
ESC.Total
ESC
PS D:\practice c++\term work c++>
```

Q6. Create a class called Time that has separate int member data for hours, minutes and seconds. One function 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 addTime() 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.

Program:

```
#include <iostream>

using namespace std;

class Time

{

private:
    int hours;

    int minutes;

    int seconds;


public:

    Time()

    {

        this->hours = 0;

        this->minutes = 0;

        this->seconds = 0;

    };

    Time(int hr, int min, int sec)

    {

        this->hours = hr;

        this->minutes = min;

        this->seconds = sec;

    };

    int getHours()

    {

        return this->hours;

    };

    int getMinutes()

    {

        return this->minutes;

    };

    int getSeconds()

    {

        return this->seconds;

    };
    void display()
    {
        cout<<"Time = " << hours << ":" << minutes << ":" << seconds << endl;
    };
    Time add(Time t1, Time t2)
    {
        int hoursAdd = t1.getHours() + t2.getHours();
        if (hoursAdd > 23)
        {
            hoursAdd -= 24;
        }
        int minutesAdd = t1.getMinutes() + t2.getMinutes();
        if (minutesAdd > 59)
        {
            minutesAdd -= 60;
            hoursAdd += 1;
        }
        int secondsAdd = t1.getSeconds() + t2.getSeconds();
        if (secondsAdd > 59)
        {
            secondsAdd -= 60;
            minutesAdd += 1;
        }
        Time t3(hoursAdd, minutesAdd, secondsAdd);
        return t3;
    };
};
int main()
{
    Time t1(11, 39, 50);

    Time t2(20, 33, 29);

    Time t3;

    t3 = t3.add(t1, t2);

    t1.display();

    t2.display();

    t3.display();

    return 0;

}
```

Output:

```
PS D:\practice c++\term work c++> cd "d:\practice c++\term work c++\" ; if ($?) { g++ q6.cpp -
o q6 } ; if ($?) { .\q6 }

Time = 11:39:50

Time = 20:33:29

Time = 8:13:19

PS D:\practice c++\term work c++>
```

Q7.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 to savingsBalance. 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 $2000.00 and $3000.00, 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.

Program:

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

class SavingsAccount
{
    float savingbal;
public:
    SavingsAccount() {}
    SavingsAccount(int value);

    static float annualInterestRate;

    void calculateMonthlyInterest();

    static void modifyIntererestRate(float value);

    float GetBalance() const
    {
        return savingbal;
    }
};
SavingsAccount::SavingsAccount(int value)
{
    savingbal = value;
}
float SavingsAccount::annualInterestRate = 0;
void SavingsAccount::calculateMonthlyInterest()
{
    savingbal = (savingbal + (savingbal * annualInterestRate) / 12);
}
void SavingsAccount::modifyIntererestRate(float value)
{
    annualInterestRate = value;
}
int main()
{
    SavingsAccount saver1(2000.00);
    SavingsAccount saver2(3000.00);
    SavingsAccount::modifyIntererestRate(4);
    saver1.calculateMonthlyInterest();
    cout << "Saver 1 Savings Balance: $" << saver1.GetBalance() << endl;
    saver2.calculateMonthlyInterest();
    cout << "Saver 2 Savings Balance: $" << saver2.GetBalance() << endl;
    cout << endl;
    SavingsAccount::modifyIntererestRate(5);
    saver1.calculateMonthlyInterest();
    cout << "Saver 1 Savings Balance: $" << saver1.GetBalance() << endl;
    saver2.calculateMonthlyInterest();
    cout << "Saver 2 Savings Balance: $" << saver2.GetBalance() << endl;
    cout << endl;
    return 0;
}
```

Output:

```
PS D:\practice c++\term work c++> cd "d:\practice c++\term work c++\" ; if ($?) { g++ q7.cpp -o q7 }
; if ($?) { .\q7 }
Saver 1 Savings Balance: $2666.67
Saver 2 Savings Balance: $4000

Saver 1 Savings Balance: $3777.78
Saver 2 Savings Balance: $5666.67

PS D:\practice c++\term work c++>
```

Q8.Define a class named Test with following description:

Data Members	
private string str	
Constructor	**Description**
Test(string)	Constructor will store value in string str.
Friend Function	**Description**
void printeven(Test ob)	This friend function will print all even position characters

Write a C++ program to print all even position characters using friend function.

Program:

```
#include <iostream>
using namespace std;
class test
{
private:
    string str;

public:
    test(string s)
    {
        str = s;
    }
    friend void printeven(test ob);
};
void printeven(test ob)
{
    int i=0;
    int l=0;
    l = ob.str.length();
    while (i < l)
    {
        if (i % 2 != 0)
        {
            cout << ob.str.at(i) << " ";
        }
        i = i + 1;
    }
}
int main()
{
    string s;
    cout << "enter the string" << endl;
    getline(cin, s);
    test ob(s);
    printeven(ob);
    return 0;
}
```

Output:

```
PS D:\practice c++\term work c++> cd "d:\practice c++\term work c++\" ; if ($?) { g++ q8.cpp -o
q8 } ; if ($?) { .\q8 }
enter the string
Abhishek
b i h k
PS D:\practice c++\term work c++>
```

Q9. Write a C++ program to create a class called OverDemo and overload teststring()
function.

void teststring(string,int)

extract the number of characters from the right side of the passing string.

void teststring(string)
Check whether passing strings is palindrome or not.

In main function invoke all member functions and perform the above task.

Program:

```
#include<iostream>

using namespace std;

class overDemo

{

    public:

    void teststring(string str, int l)

    {

        int i=0, c=0;

        for(i=l; i>=0; i--)

        {

            c++;

        }

        cout<<"Number of character from right is : "<<c<<endl;

    }

    void teststring(string h)

    {

        int i=0, l;

        string t;

        for(i=h.length()-1; i>=0; i--)

        {

            t = t + h.at(i);

        }
        if(h.compare(t)==0)

        {

            cout<<"Palindrome"<<endl;

        }

        else

        {

            cout<<"Not a Palindrome"<<endl;

        }

    }

};


int main()

{

    overDemo od;

    string s;

    int l;


    cout<<"Counting the character"<<endl;


    cout<<"Enter the string : "<<endl;

    getline(cin, s);


    l = s.length()-1;

    od.teststring(s, l);


    cout<<"\n\nFor Palindrome"<<endl;

    string t;

    cout<<"Enter the string : "<<endl;
    getline(cin, t);


    od.teststring(t);


    return 0;

}
```

Output:

```
PS D:\practice c++\term work c++> cd "d:\practice c++\term work c++\" ; if ($?) { g++ q9.cpp -
o q9 } ; if ($?) { .\q9 }

Counting the character

Enter the string :

abhishek

Number of character from right is : 8




For Palindrome

Enter the string :

abhish

Not a Palindrome

PS D:\practice c++\term work c++>
```

Q10. Create a class Complex having two int type variable named real & img denoting real and imaginary part respectively of a complex number. Overload **+ and -** operator to add, to subtract and to compare two complex numbers being denoted by the two complex type objects.

Program:

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

class Complex
```

```
{
    int real;
    int img;
    public:
    Complex(int r=0, int i=0)
    {
        real = r;
        img = i;
    }
    Complex operator+(Complex ob)
    {
        Complex res;
        res.real = real + ob.real;
        res.img = img + ob.img;
        return res;
    }
    Complex operator-(Complex ob)
    {
        Complex res;
        res.real = real - ob.real;
        res.img = img - ob.img;
        return res;
    }
    string operator==(Complex ob)
    {
        if(real == ob.real && img == ob.img)
        {
            return "True";
        }
        else
        {
            return "False";
        }
    }
    void display()
    {
        if(img<0)
        {
            img = 2*img - img;
            cout<<real<<" -i "<<img<<endl;
        }
        else if(img>1)
        {
            cout<<real<<" +i "<<img<<endl;
        }
        else if(img==1)
        {
           cout<<real<<" +i "<<img<<endl;
        }
        else if(img == 0)
        {
           cout<<real<<endl;
        }
    }
};

int main()
{
    Complex a(4, 9), b(3, 7);
    Complex c;

    c = a + b;
    c.display();

    c = a - b;
    c.display();

    string d = (a==b);
    cout<<d;

    return 0;
}
```

Output:

```
PS D:\practice c++\term work c++> cd "d:\practice c++\term work c++\" ; if ($?) { g++       q10.cpp -o
q10 } ; if ($?) { .\q10 }
            7 +i 16
            1 +i 2
            False
            PS D:\practice c++\term work c++>
```

Q11. Using the concept of operator overloading.Write a program to overload using with and without friend Function.

Unary –
Unary ++ preincrement, postincrement
Unary -- predecrement, postdecrement

With friend fuction:

Program:

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

class UnaryFriend
{
    int a = 15;
    int b = 25;

public:
    void getvalues()
    {
        cout << "Values of A & B\n";
        cout << a << "\n"<< b << "\n"<< endl;
    }
    void show()
    {
        cout << a << "\n"<< b << "\n" << endl;
    }
    void friend operator-(UnaryFriend &x);
    void friend operator++(UnaryFriend &x);
    void friend operator--(UnaryFriend &x);
};

void operator-(UnaryFriend &x)
{
    x.a = -x.a;
    x.b = -x.b;
}

void operator++(UnaryFriend &x)
{
    x.a = ++x.a;
    x.b = x.b++;
}

void operator--(UnaryFriend &x)
{
    x.a = --x.a;
    x.b = x.b--;
}

int main()
{
    UnaryFriend x1;
    UnaryFriend x2;
    UnaryFriend x3;

    x1.getvalues();

    cout << "Before Overloading\n";
    x1.show();

    cout << "After Overloading \n";
    -x1;

    x1.show();
    x2.getvalues();

    cout << "Before Pre and Post increment Overloading\n";
    x2.show();

    cout << "After Pre and Post increment Overloading \n";
    ++x2;

    x2.show();
    x3.getvalues();

    cout << "Before Pre and Post decrement Overloading\n";
    x3.show();

    cout << "After Pre and Post decrement Overloading \n";
    --x3;

    x3.show();

    return 0;
}
```

Output:

```
PS D:\practice c++\term work c++> cd "d:\practice c++\term work c++\" ; if ($?) { g++
q11_with_friend.cpp -o q11_with_friend } ; if ($?) { .\q11_with_friend }
Values of A & B
15
25

Before Overloading
15
25
```

```
After Overloading
-15
-25

Values of A & B
15
25

Before Pre and Post increment Overloading
15
25

After Pre and Post increment Overloading
16
25

Values of A & B
15
25

Before Pre and Post decrement Overloading
15
25

After Pre and Post decrement Overloading
14
25

PS D:\practice c++\term work c++>
```

Without friend function:

Program:

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

int main()
{
    int a = 1;
    cout << "a value before Unary Minus: " << a << endl;
    int b = -a;
    cout << "b value after -a : " << b << endl;
    a = 1;
    cout << "a value Post increment: " << a << endl;
    b = a++;
    cout << "b value after a++ : " << b << endl;
    cout << "a value after a++ : " << a << endl;

    a = 1;
    cout << "a value Pre increment:" << a << endl;
    b = ++a;
    cout << "b value after ++a : " << b << endl;
    cout << "a value after ++a : " << a << endl;

    a = 5;
    cout << "a value Post decrement: " << a << endl;
    b = a--;
    cout << "b value after a-- : " << b << endl;
    cout << "a value after a-- : " << a << endl;

    a = 5;
    cout << "a value Pre decrement: " << a << endl;
    b = --a;
    cout << "b value after --a : " << b << endl;
    cout << "a value after --a : " << a << endl;
    return 0;
}
```

Output:

```
PS D:\practice c++\term work c++> cd "d:\practice c++\term work c++\" ; if ($?) { g++
q11_without_friend.cpp -o q11_without_friend } ; if ($?) { .\q11_without_friend }
a value before Unary Minus: 1
b value after -a : -1
a value Post increment: 1
b value after a++ : 1
a value after a++ : 2
a value Pre increment:1
b value after ++a : 2
a value after ++a : 2
a value Post decrement: 5
b value after a-- : 5
a value after a-- : 4
a value Pre decrement: 5
b value after --a : 4
a value after --a : 4
PS D:\practice c++\term work c++>
```

Q12.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 be called?) by proper mechanism.

Program:

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

class student
{
   string name;
   int age, enroll_no, marks;

public:
   void getinfo_s()
   {
      cout << "Enter name : ";
      getline(cin, name);
      cout << "Enter age : ";
      cin >> age;
      cout << "Enter Enrollment number : ";
      cin >> enroll_no;
      cout << "Enter marks : ";
      cin >> marks;
   }
   void display()
   {
      cout << "Name : " << name << endl
          << "Age : " << age << endl
          << "Enrollment number : " << enroll_no << endl
          << "Marks : " << marks << endl;
   }
};

class faculty
{
   string fn, code, gender, deptt;
   int age, salary, experience;

public:
   void getinfo_f()
   {
      cout << "Enter faculty name : ";
      cin>>fn;
      cout << "Enter faculty code : ";
      cin >> code;
      cout << "Enter the gender : ";
      cin >> gender;
      cout << "Enter department : ";
      cin>>deptt;
      cout << "Enter age : ";
```

```
        cin >> age;
        cout << "Enter salary : ";
        cin >> salary;
        cout << "Enter experience : ";
        cin >> experience;
    }
    void display()
    {
        cout << "Faculty name : " << fn << endl
             << "Faculty code : " << code << endl
             << "Gender : " << gender << endl
             << "Department : " << deptt << endl;
        cout << "Age : " << age << endl
             << "Salary : " << salary << endl
             << "Experience : " << experience << endl;
    }
};

class person : public student, public faculty
{
};

int main()
{
    person p;

    p.getinfo_s();
    p.getinfo_f();

    cout<<"\n\nStudent Details\n\n "<<endl;
    p.student :: display();

    cout<<"\n\nFaculty Details \n\n"<<endl;
    p.faculty :: display();

    return 0;
}
```

Output:

```
PS D:\practice c++\term work c++> cd "d:\practice c++\term work c++\" ; if ($?) { g++ q12.cpp -o q12
} ; if ($?) { .\q12 }
        Enter name : Abhshek
        Enter age : 20
        Enter Enrollment number : 20021868
        Enter marks : 85
        Enter faculty name : Ashish
        Enter faculty code : pcs
        Enter the gender : male
        Enter department : cs
        Enter age : 46
        Enter salary : 800000
        Enter experience : 7
```

```
Student Details


Name : Abhshek
Age : 20
Enrollment number : 20021868
Marks : 85


Faculty Details


Faculty name : Ashish
Faculty code : pcs
Gender : male
Department : cs
Age : 46
Salary : 800000
Experience : 7
PS D:\practice c++\term work c++>
```

Q13.Implement a C++ program to demonstrate and understand Diamond problem. (Virtual base Class)

Program:

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

class student
{
   protected:
   int roll_no;
   public:
   void get_num(int a)
   {
      roll_no = a;
   }
   void put_num()
   {
      cout<<"Roll Number : "<<roll_no<<endl;
   }
};

class test : virtual public student
{
   protected:
   float m1, m2;
```

```
    public:
    void get_marks(float x, float y)
    {
        m1 = x;
        m2 = y;
    }
    void put_marks()
    {
        cout<<"\n\nMarks obtained : "<<endl;
        cout<<"Marks 1 : "<<m1<<endl<<"Marks 2 : "<<m2<<endl;
    }
};

class sports : public virtual student
{
    protected:
    float score;
    public:
    void get_score(float s)
    {
        score = s;
    }
    void put_score()
    {
        cout<<"\nSports : "<<score<<endl;
    }
};

class result : public test, public sports
{
    float total;

    public:
    void display()
    {
        total = m1 + m2 + score;

        put_num();
        put_marks();
        put_score();

        cout<<"\nTotal score : "<<total<<endl;
    }
};
```

```
int main()
{
    result r1;
    r1.get_num(678);
    r1.get_marks(46, 74);
    r1.get_score(57);

    r1.display();

    return 0;
}
```

Output:

```
PS D:\practice c++\term work c++> cd "d:\practice c++\term work c++\" ; if ($?) { g++
 q13.cpp -o q13 } ; if ($?) { .\q13 }
Roll Number : 678


Marks obtained :
Marks 1 : 46
Marks 2 : 74

Sports : 57

Total score : 177
PS D:\practice c++\term work c++>
```

Q.14 Write a C++ program to implement pure virtual function with following details:

Create A Base Class Temperature

Data members:

Float temp;
Function members

void setTempData(float)

virtual void changetemp()

Sub Class Fahrenheit (subclass of Temperature)

Data members:

Float ctemp;

Function members

Override function changetemp() to convert Fahrenheit temperature into degree Celsius by using formula C=5/9*(F-32) and display converted temperature

Sub Class Celsius (subclass of Temperature)

Data members:

Float ftemp; Function

members

Override function changetemp() to convert degree Celsius into Fahrenheit temperature by using formula F=9/5*c+32 and display converted temperature.

Program:

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

class temperature
{
protected:
   float temp;

public:
   void setdata(float a)
   {
      temp = a;
   }
   virtual void changetemp()
   {
      return;
   }
};
class fahrenheit : public temperature
{
protected:
   float ctemp;

public:
   void changetemp()
   {
      ctemp = (temp - 32) * 0.5556;
      cout << "Temperature in celsius is :" << ctemp << " C" << endl;
```

```
    }
};
class celsius : public temperature
{
protected:
    float ftemp;

public:
    void changetemp()
    {
        ftemp = (temp * 1.8) + 32;
        cout << "Temperature in Fahrenheit is : " << ftemp << " F" << endl;
    }
};
int main()
{
    temperature *ob;
    fahrenheit fob;
    float a;
    cout << "Enter the temperature in fahrenheit : " << endl;
    cin >> a;
    fob.setdata(a);
    ob = &fob;
    ob->changetemp();
    float b;
    cout << "Enter the temperature in celsius : " << endl;
    cin >> b;
    celsius cob;
    cob.setdata(b);
    ob = &cob;
    ob->changetemp();
}
```

Output:

```
PS D:\practice c++\term work c++> cd "d:\practice c++\term work c++\" ; if ($?) { g++
q14.cpp -o q14 } ; if ($?) { .\q14 }
Enter the temperature in fahrenheit :
97.5
Temperature in celsius is :36.3918 C
Enter the temperature in celsius :
37.8
Temperature in Fahrenheit is : 100.04 F
PS D:\practice c++\term work c++>
```

Q15.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)$\pi r^2 h$ Volume of hemisphere = (2/3)πr^3 Volume of cylinder = $\pi r^2 h$

Program:

```
#include<iostream>
#include <cmath>
#include <iomanip>
const float PI = 3.14;
using namespace std;
class Cal_area
{
public:
  float r, h;
  Cal_area()
  {
          r = 0;
          h = 0;
  }
  void getdata()
  {
          cout << "Enter radius and
height:" << endl;
          cin >> r >> h;
  }
  virtual void display_volume() = 0;
};
class Cone :public Cal_area
{
public: void display_volume()
{
  cout << "Volume of Cone " << (1 /
(float)3) * PI * r * r * h << endl;
}
};
class Hemisphere : public Cal_area
```

```
{
public: void display_volume()
{
  cout << "Volume of Hemisphere" <<
(2 / (float)3) * PI * r * r * r << endl;
}
};
class Cylinder : public Cal_area
{
public: void display_volume()
{
  cout << "Volume of Cylinder " <<
PI * r * r * h << endl;
}
};
int main()
{
  Cone co;
  co.getdata();
  co.display_volume();
  Hemisphere h;
  h.getdata();
  h.display_volume();
  Cylinder c;
  c.getdata();
  c.display_volume();
  return 0;
}
```

Output:

```
PS D:\practice c++\term work c++> cd
"d:\practice c++\term work c++\" ; if
($?) { g++ q15.cpp -o q15 } ; if ($?) {
.\q15 }
Enter radius and height:
5
9
Volume of Cone 235.5
```

```
Enter radius and height:
7
3
Volume of Hemisphere718.013
Enter radius and height:
5
2
Volume of Cylinder 157
PS D:\practice c++\term work c++>
```

Q16.Write a C++ to take input from a file and count number of alphabets, number of vowels and consonants.

Program:

```
#include<iostream>
#include<fstream>

using namespace std;
int main()
{
  char filename[30], ch, str[1000];
  int tot = 0, i = 0, vowels = 0, consonants = 0, alphabets = 0, ascii;
  ifstream fp;

  fp.open("C:/Users/ANCHIT SINHA/Desktop/filename.txt", ifstream::in);
  if (!fp)
  {
          cout << endl << "File doesn't exist or Access denied!";
          return 0;
  }
  while (fp >> noskipws >> ch)
  {
          str[tot] = ch;
          tot++;
  }
  fp.close();
  str[tot] = '\0';
  while (str[i] != '\0')
  {
          if ((str[i] >= 'A' && str[i] <= 'Z') || (str[i] >= 'a' && str[i] <= 'z'))
                  alphabets++;

          if (str[i] == 'a' || str[i] == 'e' || str[i] == 'i' || str[i] == 'o' || str[i] == 'u')
```

```
                    vowels++;
            else if (str[i] == 'A' || str[i] == 'E' || str[i] == 'I' || str[i] == 'O' || str[i] == 'U')
                    vowels++;
            else
            {
                    ascii = str[i];
                    if (ascii >= 65 && ascii <= 90)
                            consonants++;
                    else if (ascii >= 97 && ascii <= 122)
                            consonants++;
            }
            i++;
  }
  cout << endl << "Total Number of Alphabets = " << alphabets;
  cout << endl << "Total Number of Vowels = " << vowels;
  cout << endl << "Total Number of Consonants = " << consonants;
  cout << endl;
  return 0;
}
```

Output:

```
PS D:\practice c++\term work c++> cd
"d:\practice c++\term work c++\" ; if
($?) { g++ q16.cpp -o q16 } ; if ($?) {
.\q16 }

Total Number of Alphabets = 22
Total Number of Vowels = 8
Total Number of Consonants = 14
PS D:\practice c++\term work c++>
```