/\*

Name : Deepanshu Gupta

Section : AI & ML

University Roll No. : 2019472

1. Given the coefficients of the quadratic polynomial (float variables). Write a C++ program to determine whether the roots are real or complex (imaginary). If the roots are real, find them otherwise write the message “No real roots”.

\*/

#include<iostream>

#include<math.h>

using namespace std;

class polynomial

{

float a, b, r1, r2, y;

int c;

public:

void getdata()

{

cout<<"Enter all the coefficient:\n";

cin>>a>>b>>c;

}

void roots()

{

y=(b\*b-4\*a\*c);

if(y>0)

{

r1=(-b+pow(y,0.5))/(2\*a);

r2=(-b-pow(y,0.5))/(2\*a);

cout<<"Roots are: "<<r1<<" and "<<r2<<endl;

}

else

cout<<"No real roots\n";

}

};

int main()

{

polynomial obj;

obj.getdata();

obj.roots();

return 0;

}

\*\*\*\*\*\*OUTPUT-1\*\*\*\*\*\*

Enter all the coefficient:

1

4

5

No real roots

\*\*\*\*\*\*OUTPUT-1\*\*\*\*\*\*

Enter all the coefficient:

1

5

4

Roots are: -1 and -4

/\*

Name : Deepanshu Gupta

Section : AI & ML

University Roll No. : 2019472

2. 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. WAP to read the names of users and number of units consumed and display the charges with names.

\*/

#include<iostream>

using namespace std;

class electricity

{

float unit, cost, charge;

string name;

public:

void getdata()

{

cin.ignore();

cout<<"Enter the name of the user: ";

getline(cin, name);

cout<<"Enter the number of units consumed: ";

cin>>unit;

}

void Unit()

{

if(unit>=0 && unit<=100)

cost=60\*0.01\*unit;

else if(unit>=100 && unit<=200)

cost=80\*0.01\*unit;

else

cost=90\*0.01\*unit;

}

void Charge()

{

if(cost>300)

charge=cost+(0.15\*cost)+50;

else

charge=50+cost;

}

void display()

{

cout<<"-------------------------\n";

cout<<"Name of the user: "<<name<<endl;

cout<<"Charges of the user: "<<charge<<endl;

}

};

int main()

{

int n, i;

cout<<"Enter number of users: "; cin>>n;

electricity obj[n];

for(i=0; i<n; i++)

{

obj[i].getdata();

obj[i].Unit();

obj[i].Charge();

obj[i].display();

}

return 0;

}

\*\*\*\*\*\*OUTPUT\*\*\*\*\*\*

Enter number of users: 2

Enter the name of the user: Raj

Enter the number of units consumed: 180

-------------------------

Name of the user: Raj

Charges of the user: 194

Enter the name of the user: Ram

Enter the number of units consumed: 240

-------------------------

Name of the user: Ram

Charges of the user: 266

/\*

Name : Deepanshu Gupta

Section : AI & ML

University Roll No. : 2019472

3. W.A.P in C++ by defining a class to represent a bank account. Include the Data Members- Name of the depositor, Account number, Type of account (Saving, Current etc.), Balance amount in the account and 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

{

int accno, bal, temp;

string name, type;

public:

void initial(int x)

{

bal=x;

}

void getdata()

{

cout<<"Enter account number: ";

cin>>accno;

cin.ignore();

cout<<"Enter name of the depositor: ";

getline(cin, name);

cout<<"Enter type of account(saving or current): ";

getline(cin, type);

}

void deposit()

{

cout<<"Enter the amount to be deposited: ";

cin>>temp;

bal=bal+temp;

}

void withdraw()

{

cout<<"Enter the amount to be withdraw: ";

cin>>temp;

bal=bal-temp;

}

void display()

{

cout<<"-------------------------\n";

cout<<"Name of the depositor: "<<name<<endl;

cout<<"Balance: "<<bal<<endl;

}

};

int main()

{

int n;

cout<<"Enter your balance in your account: ";

cin>>n;

bank obj;

obj.initial(n);

obj.getdata();

obj.deposit();

obj.withdraw();

obj.display();

return 0;

}

\*\*\*\*\*\*OUTPUT\*\*\*\*\*\*

Enter your balance in your account: 150000

Enter account number: 2019472

Enter name of the depositor: Ram

Enter type of account(saving or current): Saving

Enter the amount to be deposited: 50000

Enter the amount to be withdraw: 25000

-------------------------

Name of the depositor: Ram

Balance: 175000

/\*

Name : Deepanshu Gupta

Section : AI & ML

University Roll No. : 2019472

4. W.A.P in C++ to show the working of function overloading by using a function named calculateArea() to calculate area of square, rectangle and triangle using different signatures as required.

\*/

#include<iostream>

using namespace std;

class calculator

{

int as, ar;

float ac;

public:

void calculateArea(int x)

{

as=x\*x;

cout<<"Area of square: "<<as<<endl;

}

void calculateArea(int l, int b)

{

ar=l\*b;

cout<<"Area of rectangle: "<<ar<<endl;

}

void calculateArea(float r)

{

ac=3.14\*r\*r;

cout<<"Area of circle: "<<ac<<endl;

}

};

int main()

{

int x,l,b;

float r;

calculator obj;

cout<<"Enter length of side of square: ";

cin>>x;

obj.calculateArea(x);

cout<<"Enter length and breadth of rectangle:\n";

cin>>l>>b;

obj.calculateArea(l,b);

cout<<"Enter radius of circle: ";

cin>>r;

obj.calculateArea(r);

return 0;

}

\*\*\*\*\*\*OUTPUT\*\*\*\*\*\*

Enter length of side of square: 10

Area of square: 100

Enter length and breadth of rectangle:

10

20

Area of rectangle: 200

Enter radius of circle: 7

Area of circle: 153.86

/\*

Name : Deepanshu Gupta

Section : AI & ML

University Roll No. : 2019472

5. Write a Program in C++ to demosntrate the concept of data abstraction using the concept of Class and Objects.

\*/

#include <iostream>

using namespace std;

class implementAbstraction

{

int a, b;

public:

void set(int x, int y)

{

a=x;

b=y;

}

void display()

{

cout<<"A = " <<a << endl;

cout<<"B = " << b << endl;

}

};

int main()

{

implementAbstraction obj;

obj.set(10, 20);

obj.display();

return 0;

}

\*\*\*\*\*\*OUTPUT\*\*\*\*\*\*

A = 10

B = 20

/\*

Name : Deepanshu Gupta

Section : AI & ML

University Roll No. : 2019472

6. 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 to 0.0. Write a test application named invoiceTest that demonstrates class Invoice’s capabilities.

\*/

#include<iostream>

using namespace std;

class invoice

{

string num, des;

int quantity;

double price;

public:

invoice()

{

num=" ";

des=" ";

quantity=0;

price=0;

}

void setpartnumber(string pn)

{

num=pn;

}

void setpartdescription(string pd)

{

des=pd;

}

void setquantity(int qn)

{

quantity=qn;

}

void setprice(double pr)

{

price=pr;

}

string getpartnumber()

{

return num;

}

string getpartdescription()

{

return des;

}

int getquantity()

{

return quantity;

}

double getprice()

{

return price;

}

double getinvoiceamount()

{

if(quantity<0)

quantity=0;

if(price<0)

price=0;

return quantity\*price;

}

};

int main()

{

invoice ob;

string pn,pd;

int qn;

double pr;

cout<<"Enter partnumber: ";

getline(cin,pn);

ob.setpartnumber(pn);

cout<<"Enter partdescription: ";

getline(cin,pd);

ob.setpartdescription(pd);

cout<<"Enter quantity: ";

cin>>qn;

ob.setquantity(qn);

cout<<"Enter price per item: ";

cin>>pr;

ob.setprice(pr);

cout<<"Partnumber: "<<ob.getpartnumber()<<endl;

cout<<"Partdescription: "<<ob.getpartdescription()<<endl;

cout<<"Quantity: "<<ob.getquantity()<<endl;

cout<<"Price per item: "<<ob.getprice()<<endl;

cout<<"Total amount: "<<ob.getinvoiceamount()<<endl;

return 0;

}

\*\*\*\*\*\*OUTPUT\*\*\*\*\*\*

Enter partnumber: DL1234

Enter partdescription: Manager

Enter quantity: 100

Enter price per item: 20

Partnumber: DL1234

Partdescription: Manager

Quantity: 100

Price per item: 20

Total amount: 2000

/\*

Name : Deepanshu Gupta

Section : AI & ML

University Roll No. : 2019472

7. 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>

using namespace std;

class tollbooth

{

int tc, tp, tnp;

double tm;

public:

tollbooth()

{

tp=0, tnp=0, tc=0, tm=0;

}

void payingcar()

{

tp++;

tm=tm+0.5;

}

void nonpaycar()

{

tnp++;

}

void display()

{

tc=tp+tnp;

cout<<"Total number of paying cars: "<<tp<<endl;

cout<<"Total number of non-paying cars: "<<tnp<<endl;

cout<<"Total number of cars: "<<tc<<endl;

cout<<"Total amount: "<<tm<<endl;

}

};

int main()

{

tollbooth t;

t.payingcar();

t.nonpaycar();

t.display();

return 0;

}

\*\*\*\*\*\*OUTPUT\*\*\*\*\*\*

Total number of paying cars: 1

Total number of non-paying cars: 1

Total number of cars: 2

Total amount: 0.5

/\*

Name : Deepanshu Gupta

Section : AI & ML

University Roll No. : 2019472

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

\*/

#include<iostream>

using namespace std;

class Time

{

int min, hr, sec;

public:

void input()

{

cout<<"Enter - Hour: Min: Sec\n";

cin>>hr>>min>>sec;

}

Time operator+(Time T1)

{

Time T3;

T3.sec=T1.sec+sec;

T3.min=T1.min+min;

T3.hr=T1.hr+hr;

if (T3.sec>=60)

{

T3.min++;

T3.sec=T3.sec%60;

}

if (T3.min>=60)

{

T3.hr++;

T3.min=T3.min%60;

}

return T3;

}

void display()

{

cout<<"Time = "<<hr<<":"<<min<<":"<<sec<<"\n";

}

};

int main()

{

Time T1, T2, T3;

T1.input();

T2.input();

T3=T1+T2;

T3.display();

return 0;

}

\*\*\*\*\*\*OUTPUT\*\*\*\*\*\*

Enter - Hour: Min: Sec

8

45

10

Enter - Hour: Min: Sec

5

26

13

Time = 14:11:23

/\*

Name : Deepanshu Gupta

Section : AI & ML

University Roll No. : 2019472

9. 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.

\*/

#include <iostream>

using namespace std;

class SavingsAccount

{

static float annualInterestRate;

float savingsBalance;

public:

SavingsAccount(){}

SavingsAccount(int value)

{

savingsBalance=value;

}

void calculateMonthlyInterest()

{

savingsBalance+=((savingsBalance\*annualInterestRate)/12);

}

static void modifyIntererestRate(float value);

float GetBalance() const

{

return savingsBalance;

}

};

float SavingsAccount::annualInterestRate=0;

void SavingsAccount::modifyIntererestRate(float value)

{

annualInterestRate=value;

}

int main()

{

SavingsAccount saver1(2000.00);

SavingsAccount saver2(3000.00);

SavingsAccount::modifyIntererestRate(4);

saver1.calculateMonthlyInterest();

cout<<"Balance 1: "<<saver1.GetBalance()<<endl;

saver2.calculateMonthlyInterest();

cout<<"Balance 2: "<<saver2.GetBalance()<<endl;

SavingsAccount::modifyIntererestRate(5);

saver1.calculateMonthlyInterest();

cout<<"Balance 1: "<<saver1.GetBalance()<<endl;

saver2.calculateMonthlyInterest();

cout<<"Balance 2: "<<saver2.GetBalance()<<endl;

return 0;

}

\*\*\*\*\*\*OUTPUT\*\*\*\*\*\*

Balance 1: 2666.67

Balance 2: 4000

Balance 1: 3777.78

Balance 2: 5666.67

/\*

Name : Deepanshu Gupta

Section : AI & ML

University Roll No. : 2019472

10. 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

{

int real, img;

public:

void input()

{

cout<<"Enter real and imaginary part:\n";

cin>>real>>img;

}

void show()

{

cout<<"Result = "<<real<<"+j("<<img<<")\n";

}

complex operator + (complex &C2)

{

complex C3;

C3.real=real+C2.real;

C3.img=img+C2.img;

return C3;

}

complex operator - (complex &C2)

{

complex C3;

C3.real=real-C2.real;

C3.img=img-C2.img;

return C3;

}

void operator == (complex &C2)

{

if(real==C2.real && img==C2.img)

cout<<"Yes";

else

cout<<"No";

}

};

int main()

{

complex C1, C2, C3;

C1.input();

C2.input();

C3=C1+C2;

C3.show();

C3=C1-C2;

C3.show();

C1==C2;

return 0;

}

\*\*\*\*\*\*OUTPUT\*\*\*\*\*\*

Enter real and imaginary part:

10

20

Enter real and imaginary part:

20

30

Result = 30+j(50)

Result = -10+j(-10)

No

/\*

Name : Deepanshu Gupta

Section : AI & ML

University Roll No. : 2019472

11. a. Using the concept of operator overloading with unary -.Write a program to overload using with friend Function.

\*/

#include <iostream>

using namespace std;

class abc

{

int n;

public :

void getdata()

{

cout<<"Enter value: ";

cin>>n;

}

int value()

{

return n;

}

friend abc operator-(abc &ob);

};

abc operator-(abc &ob)

{

abc ob1;

ob1.n=-ob.n;

return ob1;

}

int main()

{

abc ob;

ob.getdata();

ob=(-ob);

cout<<"New Value = "<<ob.value()<<endl;

return 0;

}

\*\*\*\*\*\*OUTPUT\*\*\*\*\*\*

Enter value: 20

New Value = -20

/\*

Name : Deepanshu Gupta

Section : AI & ML

University Roll No. : 2019472

11. b. Using the concept of operator overloading with unary -.Write a program to overload using without friend Function.

\*/

#include <iostream>

using namespace std;

class abc

{

int n;

public :

void getdata()

{

cout<<"Enter value: ";

cin>>n;

}

int value()

{

return n;

}

abc operator-()

{

abc ob1;

ob1.n=-n;

return ob1;

}

};

int main()

{

abc ob;

ob.getdata();

ob=(-ob);

cout<<"New Value = "<<ob.value()<<endl;

return 0;

}

\*\*\*\*\*\*OUTPUT\*\*\*\*\*\*

Enter value: 20

New Value = -20

/\*

Name : Deepanshu Gupta

Section : AI & ML

University Roll No. : 2019472

11. c. Using the concept of operator overloading with unary ++ preincrement. Write a program to overload using with friend Function.

\*/

#include <iostream>

using namespace std;

class abc

{

int n;

public :

void getdata()

{

cout<<"Enter value: ";

cin>>n;

}

int value()

{

return n;

}

friend abc operator++(abc &ob);

};

abc operator++(abc &ob)

{

abc ob1;

ob1.n=++ob.n;

return ob1;

}

int main()

{

abc ob, pp;

ob.getdata();

cout<<"Previous Value = "<<ob.value()<<endl;

ob=(++ob);

cout<<"New Value = "<<ob.value()<<endl;

return 0;

}

\*\*\*\*\*\*OUTPUT\*\*\*\*\*\*

Enter value: 25

Previous Value = 25

New Value = 26

/\*

Name : Deepanshu Gupta

Section : AI & ML

University Roll No. : 2019472

11. d. Using the concept of operator overloading with unary ++ preincrement. Write a program to overload using without friend Function.

\*/

#include <iostream>

using namespace std;

class abc

{

int n;

public :

void getdata()

{

cout<<"Enter value: ";

cin>>n;

}

int value()

{

return n;

}

abc operator++()

{

abc ob1;

ob1.n=++n;

return ob1;

}

};

int main()

{

abc ob, pp;

ob.getdata();

cout<<"Previous Value = "<<ob.value()<<endl;

ob=(++ob);

cout<<"New Value = "<<ob.value()<<endl;

return 0;

}

\*\*\*\*\*\*OUTPUT\*\*\*\*\*\*

Enter value: 25

Previous Value = 25

New Value = 26

/\*

Name : Deepanshu Gupta

Section : AI & ML

University Roll No. : 2019472

11. e. Using the concept of operator overloading with unary ++ postincrement. Write a program to overload using with friend Function.

\*/

#include <iostream>

using namespace std;

class abc

{

int n;

public :

void getdata()

{

cout<<"Enter value: ";

cin>>n;

}

int value()

{

return n;

}

friend abc operator++(abc &ob, int);

};

abc operator++(abc &ob, int)

{

abc ob1;

ob1.n=++ob.n;

return ob1;

}

int main()

{

abc ob, pp;

ob.getdata();

cout<<"Previous Value = "<<ob.value()<<endl;

ob=(ob++);

cout<<"New Value = "<<ob.value()<<endl;

return 0;

}

\*\*\*\*\*\*OUTPUT\*\*\*\*\*\*

Enter value: 25

Previous Value = 25

New Value = 26

/\*

Name : Deepanshu Gupta

Section : AI & ML

University Roll No. : 2019472

11. f. Using the concept of operator overloading with unary ++ postincrement. Write a program to overload using without friend Function.

\*/

#include <iostream>

using namespace std;

class abc

{

int n;

public :

void getdata()

{

cout<<"Enter value: ";

cin>>n;

}

int value()

{

return n;

}

abc operator++(int)

{

abc ob1;

ob1.n=++n;

return ob1;

}

};

int main()

{

abc ob, pp;

ob.getdata();

cout<<"Previous Value = "<<ob.value()<<endl;

ob=(ob++);

cout<<"New Value = "<<ob.value()<<endl;

return 0;

}

\*\*\*\*\*\*OUTPUT\*\*\*\*\*\*

Enter value: 25

Previous Value = 25

New Value = 26

/\*

Name : Deepanshu Gupta

Section : AI & ML

University Roll No. : 2019472

11. g. Using the concept of operator overloading with unary -- preincrement. Write a program to overload using with friend Function.

\*/

#include <iostream>

using namespace std;

class abc

{

int n;

public :

void getdata()

{

cout<<"Enter value: ";

cin>>n;

}

int value()

{

return n;

}

friend abc operator--(abc &ob);

};

abc operator--(abc &ob)

{

abc ob1;

ob1.n=--ob.n;

return ob1;

}

int main()

{

abc ob, pp;

ob.getdata();

cout<<"Previous Value = "<<ob.value()<<endl;

ob=(--ob);

cout<<"New Value = "<<ob.value()<<endl;

return 0;

}

\*\*\*\*\*\*OUTPUT\*\*\*\*\*\*

Enter value: 25

Previous Value = 25

New Value = 24

/\*

Name : Deepanshu Gupta

Section : AI & ML

University Roll No. : 2019472

11. h. Using the concept of operator overloading with unary -- preincrement. Write a program to overload using without friend Function.

\*/

#include <iostream>

using namespace std;

class abc

{

int n;

public :

void getdata()

{

cout<<"Enter value: ";

cin>>n;

}

int value()

{

return n;

}

abc operator--()

{

abc ob1;

ob1.n=--n;

return ob1;

}

};

int main()

{

abc ob, pp;

ob.getdata();

cout<<"Previous Value = "<<ob.value()<<endl;

ob=(--ob);

cout<<"New Value = "<<ob.value()<<endl;

return 0;

}

\*\*\*\*\*\*OUTPUT\*\*\*\*\*\*

Enter value: 25

Previous Value = 25

New Value = 24

/\*

Name : Deepanshu Gupta

Section : AI & ML

University Roll No. : 2019472

11. i. Using the concept of operator overloading with unary -- postincrement. Write a program to overload using with friend Function.

\*/

#include <iostream>

using namespace std;

class abc

{

int n;

public :

void getdata()

{

cout<<"Enter value: ";

cin>>n;

}

int value()

{

return n;

}

friend abc operator--(abc &ob, int);

};

abc operator--(abc &ob, int)

{

abc ob1;

ob1.n=--ob.n;

return ob1;

}

int main()

{

abc ob, pp;

ob.getdata();

cout<<"Previous Value = "<<ob.value()<<endl;

ob=(ob--);

cout<<"New Value = "<<ob.value()<<endl;

return 0;

}

\*\*\*\*\*\*OUTPUT\*\*\*\*\*\*

Enter value: 25

Previous Value = 25

New Value = 24

/\*

Name : Deepanshu Gupta

Section : AI & ML

University Roll No. : 2019472

11. j. Using the concept of operator overloading with unary -- postincrement. Write a program to overload using without friend Function.

\*/

#include <iostream>

using namespace std;

class abc

{

int n;

public :

void getdata()

{

cout<<"Enter value: ";

cin>>n;

}

int value()

{

return n;

}

abc operator--(int)

{

abc ob1;

ob1.n=--n;

return ob1;

}

};

int main()

{

abc ob, pp;

ob.getdata();

cout<<"Previous Value = "<<ob.value()<<endl;

ob=(ob--);

cout<<"New Value = "<<ob.value()<<endl;

return 0;

}

\*\*\*\*\*\*OUTPUT\*\*\*\*\*\*

Enter value: 25

Previous Value = 25

New Value = 24

/\*

Name : Deepanshu Gupta

Section : AI & ML

University Roll No. : 2019472

12. Create a Base class that consists of private, protected and public data members and member functions. Try using different access modifiers for inheriting Base class to the Derived class and create a table that summarizes the above three modes (when derived in public, protected and private modes) and shows the access specifier of the members of base class in the Derived class.

\*/

#include <iostream>

using namespace std;

class Base

{

private:

int pvt = 1;

protected:

int prot = 2;

public:

int pub = 3;

};

class Derived1: private Base

{

public:

void display()

{

cout<<"Private base private variable: Error\n";

cout<<"Private base protected variable: "<<prot<<endl;

cout<<"Private base public variable: "<<pub<<endl;

}

};

class Derived2: protected Base

{

public:

void display()

{

cout<<"Protected base private variable: Error\n";

cout<<"Protected base protected variable: "<<prot<<endl;

cout<<"Protected base public variable: "<<pub<<endl;

}

};

class Derived3: public Base

{

public:

void display()

{

cout<<"Public base private variable: Error\n";

cout<<"Public base protected variable: "<<prot<<endl;

cout<<"Public base public variable: "<<pub<<endl;

}

};

int main()

{

Derived1 obj1;

Derived2 obj2;

Derived3 obj3;

obj1.display();

obj2.display();

obj3.display();

return 0;

}

\*\*\*\*\*\*OUTPUT\*\*\*\*\*\*

Private base private variable: Error

Private base protected variable: 2

Private base public variable: 3

Protected base private variable: Error

Protected base protected variable: 2

Protected base public variable: 3

Public base private variable: Error

Public base protected variable: 2

Public base public variable: 3

/\*

Name : Deepanshu Gupta

Section : AI & ML

University Roll No. : 2019472

13. 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.

\*/

#include<iostream>

using namespace std;

class Student

{

int age;

string name, enroll\_no;

float marks;

public:

void getStudent()

{

cout<<"Enter name: ";

cin>>name;

cout<<"Enter enrollment number: ";

cin>>enroll\_no;

cout<<"Enter age: ";

cin>>age;

cout<<"Enter marks: ";

cin>>marks;

}

void display()

{

cout<<"------------------------------\n";

cout<<"Student details:\n";

cout<<"Name: "<<name<<endl;

cout<<"Enrollment Number: "<<enroll\_no<<endl;

cout<<"Age: "<<age<<endl;

cout<<"Marks: "<<marks<<endl;

}

};

class Faculty

{

string facultyName, facultyCode, deptt, gender;

float salary, experience;

int age;

public:

void getFaculty()

{

cout<<"Enter Faculty Name: ";

cin>>facultyName;

cout<<"Enter Faculty Code: ";

cin>>facultyCode;

cout<<"Enter Faculty Department: ";

cin>>deptt;

cout<<"Enter Faculty Gender: ";

cin>>gender;

cout<<"Enter Faculty Salary: ";

cin>>salary;

cout<<"Enter Faculty Age: ";

cin>>age;

cout<<"Enter Faculty Experience: ";

cin>>experience;

}

void display()

{

cout<<"------------------------------\n";

cout<<"Faculty details:\n";

cout<<"Name: "<<facultyName<<endl;

cout<<"Faclty Code: "<<facultyCode<<endl;

cout<<"Department: "<<deptt<<endl;

cout<<"Gender: "<<gender<<endl;

cout<<"Salary: "<<salary<<endl;

cout<<"Age: "<<age<<endl;

cout<<"Experience level: "<<experience<<endl;

}

};

class Person: public Student, public Faculty

{

public:

void display()

{

Student::display();

Faculty::display();

}

};

int main()

{

Person obj;

obj.getStudent();

obj.getFaculty();

obj.display();

return 0;

}

\*\*\*\*\*\*OUTPUT\*\*\*\*\*\*

Enter name: Raja

Enter enrollment number: PR101

Enter age: 20

Enter marks: 97

Enter Faculty Name: Ramu

Enter Faculty Code: CR201

Enter Faculty Department: Engineering

Enter Faculty Gender: M

Enter Faculty Salary: 150000

Enter Faculty Age: 43

Enter Faculty Experience: 20

------------------------------

Student details:

Name: Raja

Enrollment Number: PR101

Age: 20

Marks: 97

------------------------------

Faculty details:

Name: Ramu

Faclty Code: CR201

Department: Engineering

Gender: M

Salary: 150000

Age: 43

Experience level: 20

/\*

Name : Deepanshu Gupta

Section : AI & ML

University Roll No. : 2019472

14. 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>

using namespace std;

class shape

{

protected:

double x, y;

public:

void get\_data()

{

cout<<"Enter values of sides:\n";

cin>>x>>y;

}

virtual void display\_area()=0;

};

class triangle: public shape

{

public:

void display\_area()

{

cout<<"Area of triangle = "<<0.5\*x\*y<<endl;

}

};

class rectangle: public shape

{

public:

void display\_area()

{

cout<<"Area of rectangle = "<<x\*y<<endl;

}

};

int main()

{

triangle t;

rectangle r;

shape \*s[2];

s[0]=&t;

s[1]=&r;

for(int i=0;i<2;i++)

{

s[i]->get\_data();

s[i]->display\_area();

}

return 0;

}

\*\*\*\*\*\*OUTPUT\*\*\*\*\*\*

Enter values of sides:

10

20

Area of triangle = 100

Enter values of sides:

10

20

Area of rectangle = 200

/\*

Name : Deepanshu Gupta

Section : AI & ML

University Roll No. : 2019472

15. 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>

using namespace std;

class CAL\_AREA

{

public:

float r, h;

virtual float volume()=0;

void getdata(float a)

{

r=a;

}

void setdata(float b)

{

h=b;

}

};

class cone: public CAL\_AREA

{

public:

float volume()

{

return (0.33\*3.14\*r\*r\*h);

}

};

class hemisphere: public CAL\_AREA

{

public:

float volume()

{

return (0.66\*3.14\*r\*r\*r);

}

};

class cylinder: public CAL\_AREA

{

public:

float volume()

{

return (3.14\*r\*r\*h);

}

};

int main()

{

float r1,h1;

cout<<"Enter radius: ";

cin>>r1;

cout<<"Enter height: ";

cin>>h1;

cone aa;

hemisphere bb;

cylinder cc;

aa.getdata(r1);

aa.setdata(h1);

cout<<"Volume of cone: "<<aa.volume()<<" square units"<<endl;

bb.getdata(r1);

cout<<"Volume of hemisphere: "<<bb.volume()<<" square units"<<endl;

cc.getdata(r1);

cc.setdata(h1);

cout<<"Volume of cylinder: "<<cc.volume()<<" square units"<<endl;

return 0;

}

\*\*\*\*\*\*OUTPUT\*\*\*\*\*\*

Enter radius: 10

Enter height: 20

Volume of cone: 2072.4 square units

Volume of hemisphere: 2072.4 square units

Volume of cylinder: 6280 square units

/\*

Name : Deepanshu Gupta

Section : AI & ML

University Roll No. : 2019472

16. Implement a C++ program to demonstrate and understand Diamond problem.

\*/

#include<iostream>

using namespace std;

class A

{

protected:

int a;

public:

void setA()

{

cout<<"Enter A = ";

cin>>a;

}

void showA()

{

cout<<"A = "<<a<<endl;

}

};

class B:public virtual A

{

protected:

int b;

public:

void setB()

{

cout<<"Enter B = ";

cin>>b;

}

void showB()

{

cout<<"B = "<<b<<endl;

}

};

class C:public virtual A

{

protected:

int c;

public:

void setC()

{

cout<<"Enter C = ";

cin>>c;

}

void showC()

{

cout<<"C = "<<c<<endl;

}

};

class D:public B, public C

{

protected:

int d;

public:

void setD()

{

cout<<"Enter D = ";

cin>>d;

}

void showD()

{

cout<<"D = "<<d<<endl;

}

};

int main()

{

D obj;

obj.setA();

obj.setB();

obj.setC();

obj.setD();

obj.showA();

obj.showB();

obj.showC();

obj.showD();

return 0;

}

\*\*\*\*\*\*OUTPUT\*\*\*\*\*\*

Enter A = 20

Enter B = 30

Enter C = 40

Enter D = 50

A = 20

B = 30

C = 40

D = 50

/\*

Name : Deepanshu Gupta

Section : AI & ML

University Roll No. : 2019472

17. Templates are the foundation of generic programming, which involves writing code in a way that is independent of any particular type. Write a program that can create a list (create a class list) of given type (int, float, char etc.) and perform insertion and deletion on list object.

\*/

#include <iostream>

using namespace std;

#define sz 1000

template<class type>

class Array

{

public:

type dat[sz];

int bottom;

Array()

{

bottom=-1;

}

void Insert(type item)

{

dat[++bottom]=item;

}

type Delete()

{

if(bottom==-1)

return 0;

return dat[bottom--];

}

void show()

{

for(int i=0; i<=bottom; i++)

cout<<dat[i]<<endl;

}

};

int main()

{

Array<int>ob;

ob.Insert(20);

ob.Insert(10);

ob.Insert(30);

ob.Insert(40);

cout<<"All items:\n";

ob.show();

cout<<"Element deleted:"<<ob.Delete()<<endl;

cout<<"All items:\n";

ob.show();

return 0;

}

\*\*\*\*\*\*OUTPUT\*\*\*\*\*\*

All items:

20

10

30

40

Element deleted:40

All items:

20

10

30

/\*

Name : Deepanshu Gupta

Section : AI & ML

University Roll No. : 2019472

18. Write a C++ program to implement different methods of List, Vector and Map in STL (Standard Template Library).

\*/

#include<iostream>

#include<list>

#include<vector>

#include<map>

using namespace std;

int main()

{

list<int>myList;

myList.push\_back(1);

myList.push\_back(2);

myList.push\_back(3);

myList.push\_front(0);

cout<<"Elements in myList: ";

for (int x: myList)

cout<<x<<"\t";

cout<<endl;

vector<string>myVector;

myVector.push\_back("apple");

myVector.push\_back("banana");

myVector.push\_back("orange");

cout<<"Second element in myVector: "<<myVector[1]<<endl;

cout<<"Third element in myVector: "<<myVector.at(2)<<endl;

map<int, string>myMap;

myMap[1]="one";

myMap[2]="two";

myMap[3]="three";

cout<<"Value of key 2 in myMap: "<<myMap[2]<<endl;

if (myMap.count(4)>0)

cout<<"Key 4 exists in myMap" <<endl;

else

cout<<"Key 4 does not exist in myMap"<<endl;

return 0;

}

\*\*\*\*\*\*OUTPUT\*\*\*\*\*\*

Elements in myList: 0 1 2 3

Second element in myVector: banana

Third element in myVector: orange

Value of key 2 in myMap: two

Key 4 does not exist in myMap