```cpp
/*
Aryan 7070
Program-27
Unary Operator Overloading
*/
#include<iostream>
using namespace std;

class space{
    int x;
    int y;
    int z;
    public:
    void getdata(int a,int b,int c);
    void display(void);
    void operator-();
};
void space :: getdata(int a,int b,int c){
    x=a;
    y=b;
    z=c;
}
void space :: display(void){
    cout << "x = " << x << " ";
    cout << "Y = " << y << " ";
    cout << "Z =" << z << " ";
}
void space :: operator-(){
    x = -x;
    y = -y;
    z = -z;
}
int main(){
    space S;
    S.getdata(10,-20,30);
    cout << "S : ";
    S.display();

    -S;
    cout << "\n-S : ";
    S.display();

    return 0;
}
```

Output :

S : x = 10 Y = -20 Z =30

-S : x = -10 Y = 20 Z =-30

```
/*
Aryan 7070
Program-28
Binary Operator Overloading
*/
#include <iostream>
using namespace std;
class complex
{
    float real, imag;

public:
    complex()
    {
        real = imag = 0;
    }
    void read()
    {
        cout << "Enter the numbers : " << endl;
        cin >> real >> imag;
    }
    complex operator+(complex a)
    {
        complex temp;
        temp.real = real + a.real;
        temp.imag = imag + a.imag;
        return temp;
    }
    void show()
    {
        if (imag > 0)
            cout << real << "+"
                 << "i" << imag;
    }
};
int main()
{
    complex c1, c2, c3;
    c1.read();
    c2.read();
    c3 = c1 + c2;
    cout << "\nC1 : ";c1.show();
    cout << "\nC2 : ";c2.show();
    cout << "\nC3 : ";c3.show();
    return 0;
```

```
}
```

Output :

```
Enter the numbers :
12
12
Enter the numbers :
10
10

C1 : 12+i12
C2 : 10+i10
C3 : 22+i22
```

```
/*
Aryan 7070
Program 29 : Conversion from user to basic
*/
#include<iostream>
using namespace std;

class sample{
    int hrs, min;
    public:
    sample(int x,int y){
        hrs = x;
        min = y;
    }
    operator int(){
        return(hrs*60+min);
    }
};
int main(){
    int h,m,duration;
    h=2,m=40;
    sample s(h,m);
    duration=s;
    cout << duration;
    }
```

Output :

160

```
/*
Aryan 7070
Program 30 : Conversion from basic to userdefined
*/
#include<iostream>
using namespace std;

class sample{
    int hours,mins;
    public:
    sample(){
        hours = 0;
        mins = 0;
    }
    void display(){
        cout << hours << " "  << mins;
    }
    sample(int t){
        hours = t/60;
        mins = t%60;
    }
};
int main(){
    sample s;
    int x = 200;
    s = x;
    s.display();
    return 0;
}
```

Output :

3 20

```
/*
Aryan 7070
Program 31 : Conversion from user defined to userdefined
*/
#include<iostream>
using namespace std;
class product
{
    public:
    int m, n;

    void setdata(int x, int y)
    {
        m=x;
        n=y;
    }
};
class item
{
    private:
    int a, b;
    public:
    void showdata()
    {
        cout<<a<<b;
    }
    item()
    {
        a=b=0;
    }
    item(product P)
    {
        a=P.m;
        b=P.n;
    }
};

int main()
{
```

```
    item i1;
    product p1;
    p1.setdata(3,5);
    i1=p1;
    i1.showdata();
    return 0;
}
```

Output :

```
35
```

```
/*
Aryan 7070
Program-32 Single Inheritance
*/
#include<iostream>
using namespace std;

class B{
    int a;
    public:
    int b;
    void set_ab();
    int get_a(void);
    void show_a(void);
};
class D : public B{
    int c;
    public:
    void mul(void);
    void display(void);
};
void B :: set_ab(void){
    a = 5;
    b = 10;
}
int B :: get_a(){
   return a;
}
void B :: show_a(){
    cout << "a = " << a << "\n";
}
void D :: mul(){
    c = b*get_a();
}
void D :: display(){
    cout << "a = " << get_a() << "\n";
    cout << "b = " << b << "\n";
    cout << "c = " << c << "\n";
}
int main(){
    D d;
    d.set_ab();
```

```
    d.mul();
    d.show_a();
    d.display();
    d.b = 20;
    d.mul();
    d.display();
    return 0;
}
```

Output :

a = 5

a = 5

b = 10

c = 50

a = 5

b = 20

c = 100

```
/*
Aryan 7070
Program-33 , Multilevel inheritance
*/
#include <iostream>
using namespace std;

class student{
    protected:
    int roll_number;
    public:
    void get_number(int);
    void put_number(void);
};
void student :: get_number(int a){
    roll_number = a;
}
void student :: put_number(){
    cout << "Roll Number: " << roll_number << "\n";
}
class test : public student{
    protected:
    float sub1;
    float sub2;
    public:
    void get_marks(float, float);
    void put_marks(void);
};
void test :: get_marks(float x,float y){
    sub1 = x;
    sub2 = y;
}
void test :: put_marks(){
    cout << "Marks in Sub1 = " << sub1 << "\n";
    cout << "Marks in Sub2 = " << sub2 << "\n";
}
class result : public test{
    float total;
    public:
    void display(void);
};
void result :: display(void){
    total =  sub1 +sub2;
```

```
    put_number();
    put_marks();
    cout << "Total = " << total << endl;
}
int main(){
    result student1;
    student1.get_number(111);
    student1.get_marks(75.0,59.5);
    student1.display();
    return 0;
}
```

Output :

Roll Number: 111

Marks in Sub1 = 75

Marks in Sub2 = 59.5

Total = 134.5

```
/*
Aryan 7070
Program 34 : Multiple Inheritance
*/
#include<iostream>
using namespace std;

class M{
    protected:
    int m;
    public:
    void get_m(int);
};
class N{
    protected:
    int n;
    public:
    void get_n(int);
};
class P : public M, public N{
    public:
    void display(void);
};
void M :: get_m(int x){
    m = x;
}
void N :: get_n(int y){
    n = y;
}
void P :: display(void){
    cout << "m = " << m << "\n";
    cout << "n = " << n << "\n";
    cout << "m*n =" << m*n << "\n";
}
int main(){
    P p;
    p.get_m(10);
    p.get_n(20);
    p.display();
    return 0;
}
```

Output :

m = 10

n = 20

m*n =200

```
/*
Aryan 7070
Program-35 Program to resolve inheritance ambiguity
*/
#include<iostream>
using namespace std;

// Base class A

class A {
    public:

    void func() {
        cout << " I am in class A" << endl;
    }
};

// Base class B

class B {
    public:

    void func() {
        cout << " I am in class B" << endl;
    }
};
// Derived class C
class C: public A, public B {
};

int main() {
    C obj;
 // Calling function func() in class A
    obj.A::func();

    // Calling function func() in class B
    obj.B::func();

    return 0;
}
```

Output :

I am in class A

I am in class B

```
/*
Aryan 7070
Program 36 : Hybrid Inheritance
*/
#include<iostream>
using namespace std;

class student{
    protected:
    int roll_number;
    public:
    void get_number(int a){
        roll_number = a;
    }
    void put_number(){
        cout << "Roll no " << roll_number << endl;
    }
};
class test : public student{
    protected:
    float part1, part2;
    public:
    void get_marks(float x,float y){
        part1 = x;
        part2 = y;
    }
    void put_marks(){
        cout << "Marks Obtained: " << endl;
        cout << "Part1 =  " << part1 << endl;
        cout << "Part2 = " << part2 << endl;
    }
};
class sports{
    protected:
    float score;
    public:
    void get_score(float s){
        score = s;
    }
    void put_score(){
        cout << "Sports wt : " << score << "\n\n";
    }
};
class result : public test, public sports{
    float total;
    public:
    void display();
};
void result :: display(){
    total = part1 + part2 + score;
```

```
    put_number();
    put_marks();
    put_score();
    cout << "Total score : " << total << endl;
}
int main(){
    result student_1;
    student_1.get_number(1234);
    student_1.get_marks(27.5,33.0);
    student_1.get_score(6.0);
    student_1.display();
    return 0;
}
```

Output :

Roll no 1234

Marks Obtained:

Part1 = 27.5

Part2 = 33

Sports wt : 6

Total score : 66.5

```
/*
Aryan 7070
Program: 37 Virtual base class
*/
#include<iostream>
using namespace std;
class student{
    protected:
    int roll_number;
    public:
    void get_number(int a){
        roll_number = a;
    }
    void put_number(){
        cout << "Roll no " << roll_number << endl;
    }
};
class test : virtual public student{
    protected:
    float part1,part2;
    public:
    void get_marks(float x,float y){
        part1 = x;
        part2 = y;
    }
    void put_marks(){
        cout << "Marks obtained : " << endl;
        cout << "Part1 = " << part1 <<endl;
        cout << "Part2 = " << part2 <<endl;
    }
};
class sports : public virtual student{
    protected:
    float score;
    public:
    void get_score(float s){
        score = s;
    }
    void put_score(){
        cout << "Sports wt: " << score << "\n\n";
    }
};
class result : public test, public sports{
    float total;
    public:
```

```
    void display();
};
void result :: display(){
    total = part1 + part2 + score;

    put_number();
    put_marks();
    put_score();
    cout << "Total score : " << total << endl;
}
int main(){
    result student_1;
    student_1.get_number(678);
    student_1.get_marks(30.5,25.5);
    student_1.get_score(7.0);
    student_1.display();
    return 0;
}
```

Output :

Roll no 678

Marks obtained :

Part1 = 30.5

Part2 = 25.5

Sports wt: 7

Total score : 63

```
/*
Aryan 7070
Program-38 : Pointer to derived class
*/
#include <iostream>
using namespace std;

class base{
public:
 void display(){
cout<<"Base class display called\n";
}
};

class derv1 : public base{
public:
void display(){
cout<<"Derv1's display called\n";
}
};

class derv2 : public base {
public :
void display(){
cout<<"Derv2's display called\n";
}
};


int main(){
base *ptr; // pointer to base class

derv1 d1; // derived (derv1) object

derv2 d2;

ptr =&d1; // address of d1 to base pointer

ptr->display();

ptr=&d2; // address of d2 to base pointer

ptr->display();
return 0;
}
```

Output :

Base class display called

Base class display called

```
/*
Aryan 7070
Program-39 : Pointer to derived class using virtual functiuon
*/
#include <iostream>
using namespace std;

class base{
public:
 virtual void display(){
cout<<"Base class display called\n";
}
};

class derv1 : public base{
public:
void display(){
cout<<"Derv1's display called\n";
}
};

class derv2 : public base {
public :
void display(){
cout<<"Derv2's display called\n";
}
};


int main(){
base *ptr; // pointer to base class

derv1 d1; // derived (derv1) object

derv2 d2;

ptr =&d1; // address of d1 to base pointer

ptr->display();

ptr=&d2; // address of d2 to base pointer

ptr->display();
return 0;
```

```
}
```

Output :

```
Derv1's display called
Derv2's display called
```

```
/*
Aryan 7070
Program-40
Exception handling : Rethrow an exception
*/
#include<iostream>
#include<conio.h>
using namespace std;

void funchandler(){
    try{
        throw 10;
    }
        catch(int i){
            cout << "Caught Exception inside function\n";
            throw; //rethrow
        }
    }
int main(){
    cout << "Start of main() \n";
    try {
        funchandler();
    }
    catch(int i){
        cout << "Rethrown exception caught in main()\n";
    }
    cout << "End of main()";
    return 0;
}
```

Output :

Start of main()

Caught Exception inside function

Rethrown exception caught in main()

End of main()

```
/*
Aryan 7070
Program-41
Exception handling : Program to check whether a person is eligible to vote or not
*/
#include<iostream>
#include<conio.h>
using namespace std;

int main(){
    int age;
    cout << "Enter age for voting(18 to 120) : ";
    cin>> age;
    try{
        if(age>0 && age<18){
            throw 0 ;
        }
        else if(age>120){
            throw 'v';
        }
        else if(age<0){
            throw 2.8;
        }
        cout << "Eligible for voting";
    }
    catch(int i){
        cout << "Exception : Valid age but not eligible for voting";
    }
    catch(...){
        cout << "exception : Invalid age for voting";
    }
    return 0;
}
```

Output :

Enter age for voting(18 to 120) : 12

Exception : Valid age but not eligible for voting

```
/*
Aryan 7070
Program-42
Exception handling : To calculate square root
*/
#include<iostream>
#include<math.h>
using namespace std;

int main(){
    int num;
    double res;
    cout << "Enter a number : ";
    cin >> num;
    try {
        if(num<0)
            throw 10;
        else if(num>0)
            throw 'E';
        cout << "Square root of " << num << " is " << sqrt(num);
    }
    catch(int){
        cout << "Exception handling : out of range \n ";
    }
    catch(char){
        cout << "Exception : square root of negative number doesn't exist";
    }
    return 0;
}
```

Output :

Enter a number : -1

Exception handling : out of range

```
/*
Aryan 7070
Program-43
Exception handling
*/
#include<iostream>
using namespace std;

int main(){
    int x,y;
    cout << "Enter numerator (x) and denominator (y)  = ";
    cin >> x >> y;
    try{
        if(y==0)
        throw 10;
        cout << "x / y = " << x / y;
    }
    catch(int i){
        cout << "Exception : Division by 0 not allowed";
    }
    return 0;
}
```

Output :

Enter numerator (x) and denominator (y) = 12

0

Exception : Division by 0 not allowed