**1]Function Call Bracket Overload:**

#include<iostream>

class Demo

{

    public:

        int a,b,c;

    Demo& operator()(int x)

    {

        std::cout<<"Hello World....!"<<std::endl;

    }

    Demo& operator()(int a,int b)

    {

        std::cout<<a+b<<std::endl;

        return \*this;

    }

    Demo& operator()(int a,int b,int c)

    {

        std::cout<<a+b+c;

        return \*this;

    }

};

int main()

{

    Demo d;

    d(10,20)(1)(10,20,30);   //d.operator()(10,20)

}

**2]Logical And Operator Overload:**

#include<iostream>

class Demo

{

    public:

        int x,y;

    Demo(int x,int y)

    {

        this->x=x;

        this->y=y;

    }

    bool operator&&(const Demo& ref)

    {

        return this->x&&ref.x && this->y&&ref.y;

    }

};

int main()

{

    Demo d1(10,20);

    Demo d2(30,40);

    //Demo d2(40,0);

    if(d1&&d2)  //d1.operator&&(d2)

    {

        std::cout<<"Result(d1&&d2):"<<"1";

    }

    else

    {

        std::cout<<"Result(d1&&d2):"<<"0";

    }

}

**3]Logical OR Operator Overload:**

#include<iostream>

class Demo

{

    public:

        int x,y;

    Demo(int x,int y)

    {

        this->x=x;

        this->y=y;

    }

    bool operator||(const Demo& ref)

    {

        return this->x||ref.x || this->y||ref.y;

    }

};

int main()

{

    Demo d1(10,20);

    Demo d2(30,40);

    //Demo d2(40,0);

    if(d1||d2)  //d1.operator&&(d2)

    {

        std::cout<<"Result(d1||d2):"<<"1";

    }

    else

    {

        std::cout<<"Result(d1||d2):"<<"0";

    }

}