PROGRAM-

CREATE A CLASS PHONE NUMBER WITH DATA MEMBERS – AREA CODE, EXCHANGE, LINE. USING INSERTION AND EXTRACTION OPERATOR OVERLOADING, GET VALUE AND PRINT.

//BHAVNA VERMA- 171210019-29/01/2019

//OVERLOADING OF INSERTION AND EXTRACTION OPERATOR

```
#include<iostream>
using namespace std;
class phone_number
{
        string areacode;
        string exchange;
        string line;
        public:
        friend ostream & operator << (ostream &out, const phone_number &p);
   friend istream & operator >> (istream &in,  phone_number &p);
};
ostream & operator << (ostream &out, const phone_number &p)
{
   out << "(" <<p.areacode << ") -" << p.exchange << p.line;
   return out;
}

istream & operator >> (istream &in,  phone_number &p)
{
   in >> p.areacode;
        in >> p.exchange ;
        in >> p.line;
```

```
    return in;
}


int main()
{
   phone_number p1;
   cout<<"ENTER THE PHONE NUMBER IN FORMAT (AREA CODE <SPACE> EXCHANGE <SPACE> LINE"<<endl;
   cin >> p1;
   cout << p1;
   return 0;
}
```

```
ENTER THE PHONE NUMBER IN FORMAT (AREA CODE <SPACE> EXCHANGE <SPACE> LINE

123 45 67890
(123) -4567890
--------------------------------
Process exited after 6.848 seconds with return value 0
Press any key to continue . . .
```

OVERLOADING OF INSERTION AND EXTRACTION OPERATOR –

In C++, stream insertion operator “<<” is used for output and extraction operator “>>” is used for input.
1) cout is an object of ostream class and cin is an object istream class
2) These operators must be overloaded as a global function. And if we want to allow them to access private data members of class, we must make them friend.

The operators ‘<<' and '>>' are called like 'cout << ob1' and 'cin >> ob1'. So if we want to make them a member method, then they must be made members of ostream and istream classes, which is not a good option most of the time. Therefore, these operators are overloaded as global functions with two parameters, cout and object of user defined class.

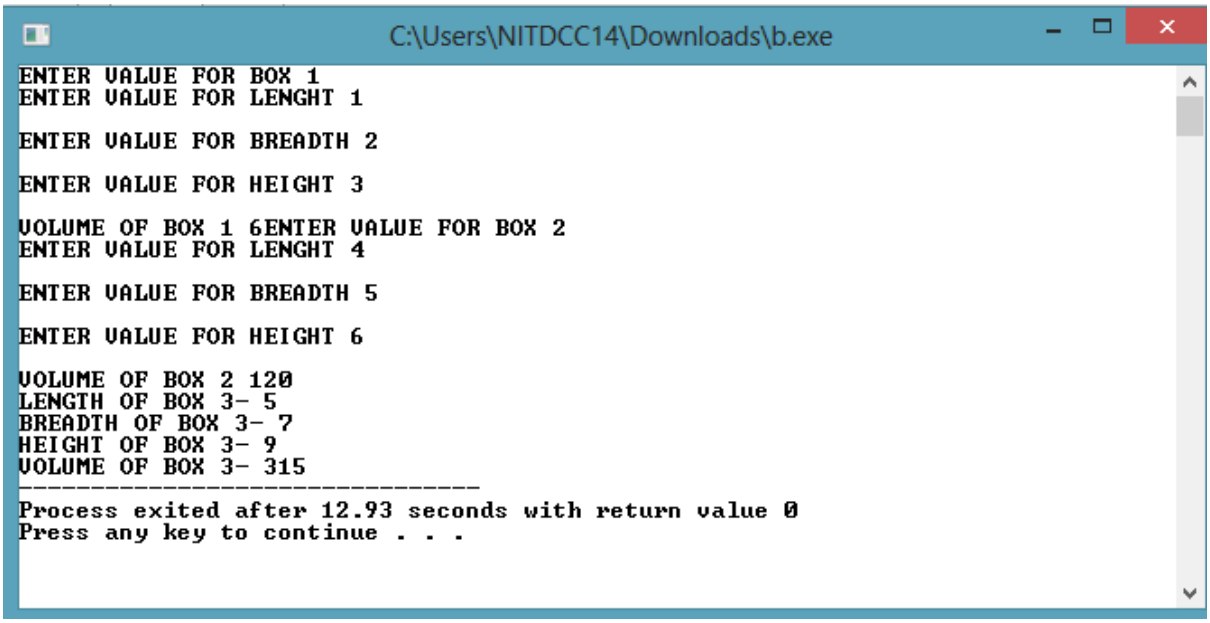
PROGRAM-

CREATE A CLASS BOX WITH DATA MEMBERS – LENGTH, BREADTH, HEIGHT, VOLUME. USING + OPERATOR OVERLOADING, GET VALUES FOR TWO OBJECTS OF BOX AND GET THE SUM OF THESE VALUES FOR THIRD OBJECT AND PRINT.

```
//BHAVNA VERMA-171210019-29/01/2019

//CLASS BOX OPERATOR OVERLOADING OF + AND -

#include<iostream>

using namespace std;

class box

{

        private:

        int length;

        int breadth;

        int height;

        int vol;

        public:

        void getvalue( box &o1, box &o2)

        {

        int v1, v2;

        cout<<"ENTER VALUE FOR BOX 1"<<endl;

        cout<<"ENTER VALUE FOR LENGHT ";

        cin>>o1.length;

        cout<<"\nENTER VALUE FOR BREADTH ";

        cin>>o1.breadth;

        cout<<"\nENTER VALUE FOR HEIGHT ";

        cin>>o1.height;

        v1=o1.length*o1.breadth*o1.height;
```

```
        cout<<"\nVOLUME OF BOX 1 "<<v1;
        cout<<"ENTER VALUE FOR BOX 2"<<endl;
        cout<<"ENTER VALUE FOR LENGHT ";
        cin>>o2.length;
        cout<<"\nENTER VALUE FOR BREADTH ";
        cin>>o2.breadth;
        cout<<"\nENTER VALUE FOR HEIGHT ";
        cin>>o2.height;
        v2=o2.length*o2.breadth*o2.height;
        cout<<"\nVOLUME OF BOX 2 "<<v2;
   };
        friend box operator + (box const &o1, box const &o2 ) ;
        friend print( box b1);
};

box operator + (box const &o1, box const &o2 )
{
   box o3;
   o3.length=o1.length+o2.length;
   o3.breadth=o1.breadth+o2.breadth;
        o3.height=o1.height+o2.height;
        o3.vol=o3.length*o3.breadth*o3.height;
   return o3;
}

print( box b1)
```

```
{
        cout<<"\nLENGTH OF BOX 3- "<<b1.length;
        cout<<"\nBREADTH OF BOX 3- "<<b1.breadth;
        cout<<"\nHEIGHT OF BOX 3- "<<b1.height;
        cout<<"\nVOLUME OF BOX 3- "<<b1.vol;
}
int main()
{
        box b1, b2, b3;
        b1.getvalue(b1, b2);
        b3=b1+b2;
        print(b3);
        return 0;
}
```



+ OPERATOR OVERLOADING

In C++, we can make operators to work for user defined classes. This means C++ has the ability to provide the operators with a special meaning for a data type, this ability is known as operator overloading.

For example, we can overload an operator ‘+’ in a class like String so that we can concatenate two strings by just using +.
Other example classes where arithmetic operators may be overloaded are Complex Number, Fractional Number, Big Integer, etc.