<u>PROGRAM</u> – CREATE A CLASS INVENT1 WITH DATA MEMBER – INT CODE, ITEMS; FLOAT PRICE; FUNCTIONS- CONSTRUCTOR, FUNCTION TO RETURN CODE, ITEMS, PRICE, DISPLAY DATA; ***USE OPERATOR TO CONVERT INVENT1 TO INVENT2 INSIDE CLASS INVENT1.*** CREATE CLASS INVENT2 WITH DATA MEMBERS – INT CODE; FLOAT VALUE; FUNCTIONS- CONSTRUCTOR, FUNCTION TO RETURN CODE, VALUE, DISPLAY DATA, ***USE TYPE CONVERSION TO CONVERT INVENT1 TO INVENT2 INSIDE CLASS INVENT2***

//BHAVNA VERMA-171210019-19/02/2019

//TYPE CONVERSION USING OVERLOADED CASTING OPERATOR

```
#include<iostream>

using namespace std;

class invent2 //DESTINATION CLASS

{

	public:

		int code;

		float value;


		invent2() //DEFAULT CONSTRUCTOR

		{

			code=0;

			value=0;

		}

		void putdata() //FUNCTION TO PRINT DATA

		{

			cout<<"code = "<<code<<endl;

			cout<<"Value = "<<value<<endl;

		}

		int getcode()

		{
```

```
                        return code;
                }
                int getvalue()
                {
                        return value;
                }
};
class invent1 //SOURCE CLASS
{
        int code;
        int items;
        float price;
        public:
                invent1() //CONSTRUCTOR
                {
                        code=2;
                        items=5;
                        price=60.1;
                }
                void putdata() //FUNCTION TO PRINT DATA
                {
                        cout<<"code = "<<code<<endl;
                        cout<<"items = "<<items<<endl;
                        cout<<"price = "<<price<<endl;
                }
                int getcode()
                {
```

```
        return code;
    }
    int getitems()
    {
        return items;
    }
    float getprice()
    {
        return price;
    }
    //CLASS OBJECT TO BASIC TYPE - INVENT1 OBJECT CONVERTED
TO FLOAT TYPE
    operator float() //OVERLOADED CASTING OPERATOR
    {
        return float(items*price);
    }
    //ONE CLASS TO ANOTHER CLASS TYPE - INVENT1 CONVERTED
TO INVENT2 TYPE
    operator invent2() //OVERLOADED CASTING OPERATOR
    {
        invent2 temp;
        temp.code=code;
        temp.value=price*items;
        return temp;
    }
};
```

```
int main()
{
        float total_price;
        invent1 o1;
        cout<<"\nValues in object of invent1 class \n";
        o1.putdata();
        //CONVERSION FROM CLASS TO BASIC TYPE (INVENT1 TO
TOTALPRICE_(FLOAT) TYPE)
        total_price=o1;
        cout<<"\nAfter type conversion of invent1 to float type total_price are as follows \n";
        cout<<"\nTotal price = "<<total_price;
        //CONVERSION FROM ONE CLASS TO ANOTHER CLASS TYPE (INVENT1
TO INVENT2 TYPE)
        invent2 o2;
        cout<<"\n\nValues in object of invent2 class \n ";
        o2.putdata();
        o2=o1;
        cout<<"\nAfter type conversion of invent1 to invent2 -> values in invent2 are as
follows \n";
        o2.putdata();
        return 0;
}
```

//TYPE CONVERSION USING CONSTRUCTOR

```
#include<iostream>
using namespace std;
class invent1 //SOURCE CLASS
{
        int code;
}

int items;
float price;
public:
        invent1() //CONSTRUCTOR
        {
                code=2;
                items=5;
                price=60.1;
        }
        void putdata() //FUNCTION TO PRINT DATA
        {
                cout<<"code = "<<code<<endl;
                cout<<"items = "<<items<<endl;
                cout<<"price = "<<price<<endl;
        }
        int getcode()
        {
                return code;
        }
        int getitems()
        {
                return items;
        }
        float getprice()
        {
                return price;
        }
```

```
		//CLASS OBJECT TO BASIC TYPE - INVENT1 OBJECT CONVERTED
TO FLOAT TYPE
		operator float() //OVERLOADED CASTING OPERATOR
		{
			return float(items*price);
		}
};
class invent2 //DESTINATION CLASS
{
	int code;
	float value;
	public:
		invent2() //DEFAULT CONSTRUCTOR
			{
				code=0;
				value=0;
			}
			void putdata() //FUNCTION TO PRINT DATA
			{
				cout<<"code = "<<code<<endl;
				cout<<"Value = "<<value<<endl;
			}
			int getcode()
			{
				return code;
			}
			int getvalue()
```

```
                        {
                                return value;
                        }
                        //ONE CLASS TO ANOTHER CLASS TYPE - INVENT1
CONVERTED TO INVENT2 TYPE USING CONSTRUCTOR
                        invent2(invent1 p)
                        {
                                code=p.getcode();
                                value=p.getitems()*p.getprice();
                        }
};
int main()
{
        float total_price;
        invent1 o1;
        cout<<"\nValues in object of invent1 class \n";
        o1.putdata();
        //CONVERSION FROM CLASS TO BASIC TYPE (INVENT1 TO
TOTALPRICE_(FLOAT) TYPE)
        total_price=o1;
        cout<<"\nAfter type conversion of invent1 to float type total_price are as follows \n";
        cout<<"\nTotal price = "<<total_price;
        //CONVERSION FROM ONE CLASS TO ANOTHER CLASS TYPE (INVENT1
TO INVENT2 TYPE)
        invent2 o2;
        cout<<"\n\nValues in object of invent2 class \n ";
        o2.putdata();
        o2=o1;
```

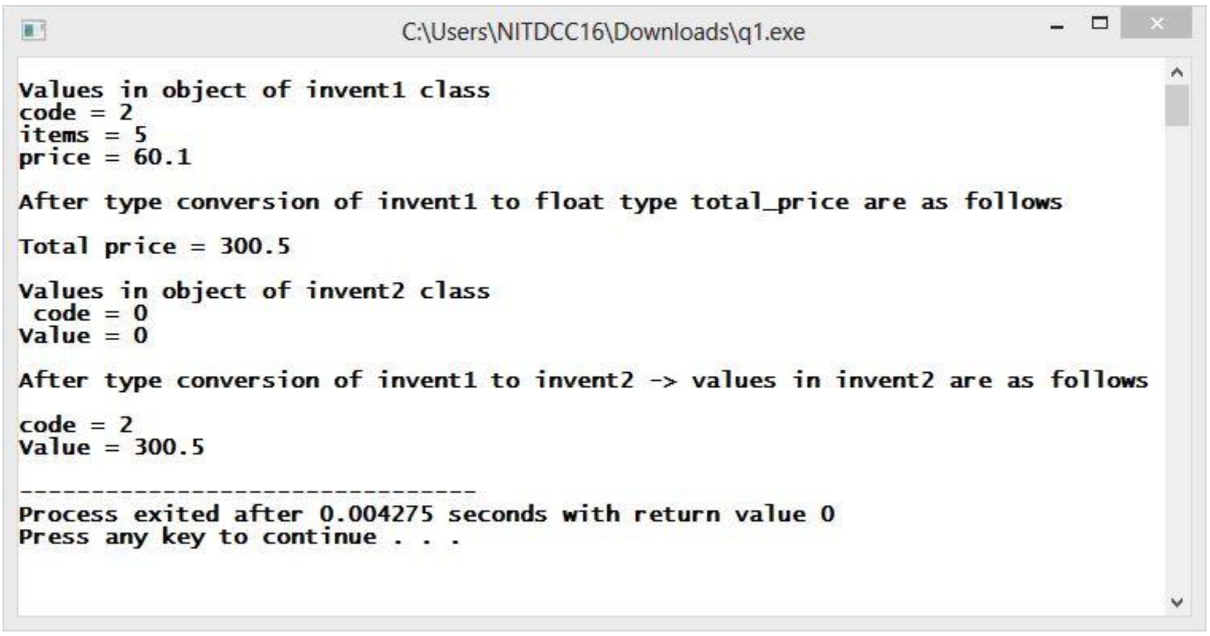
```
	cout<<"\nAfter type conversion of invent1 to invent2 -> values in invent2 are as follows \n";

	o2.putdata();

	return 0;

}
```

OUTPUT



```
Values in object of invent1 class
code = 2
items = 5
price = 60.1

After type conversion of invent1 to float type total_price are as follows

Total price = 300.5

Values in object of invent2 class
 code = 0
Value = 0

After type conversion of invent1 to invent2 -> values in invent2 are as follows

code = 2
Value = 300.5

--------------------------------
Process exited after 0.004275 seconds with return value 0
Press any key to continue . . .
```

TYPE CONVERSION –

I. AUTOMATIC TYPE CONVERSION is performed by compiler implicitly when different types of constant or literals are used in mixed expression.

II. FOR TYPE CONVERSION OF

1) BASIC DATA TYPE TO CLASS -> using constructor in class

2) A CLASS TO BASIC DATA TYPE -> using overloaded casting operator

3) ONE CLASS TO ANOTHER CLASS TYPE -> either using constructor (in destination class) or overloaded casting operator(in source class) as in above programs