Type Casting and Conversion

1.Write a C++ program to convert Primitive type to Complex type.
Example -

```
int main()
{
Complex c1;
Int x=5;
c1=x;
return 0;
}
```

```cpp
#include <iostream>
using namespace std;

class Complex
{
private:
    int real, img;

public:
    Complex()
    {
        real = 0;
        img = 0;
    }
    Complex(int x)
    {
        real = x;
        img = x;
    }
    void display()
    {
        cout << "real = " << real << " img = " << img << endl;
    }
};

int main()
{
    Complex c1;
    int x = 5;
    c1.display();
    c1 = x;
    c1.display();
    return 0;
}
```

```
==========
Output:
real = 0 img = 0
real = 5 img = 5
```

2. Write a C++ program to convert Complex type to Primitive type.
Example -

```
int main()
{
```

```
Complex c1;
c1.setData(3,4);
int x;
x=c1;
return 0;
}
```

```cpp
#include <iostream>
using namespace std;

class Complex
{
private:
    int real, img;

public:
    Complex()
    {
        real = 0;
        img = 0;
    }
    void setData(int x, int y)
    {
        real = x;
        img = y;
    }
    operator int()
    {
        return (real + img);
    }
    void display()
    {
        cout << "real = " << real << " img = " << img << endl;
    }
};

int main()
{
    Complex c1;
    c1.setData(3, 4);
    int x;
    x = c1;
    c1.display();
    cout << "real + img = " << x << endl;
    return 0;
```

```
}
================================================================================
Output:
real = 3 img = 4
real + img = 7
```

3. Create a Product class and convert Product type to Item type using constructor

```
int main()
{
Item i1;
Product p1;
p1.setData(3,4);
i1=p1;
return 0;
}
```

```
#include <iostream>
using namespace std;

class Product
{
    private:
        int a, b;
    public:
        Product(){}
        Product(int x, int y)
        {
            a = x;
            b = y;
        }
        int getData()
        {
            return(a*b);
        }
        void showData()
        {
            cout << "a = " << a << " b = " << b << endl;
        }
};

class Item
{
    private:
        int a;
    public:
        Item(){}
        Item(Product p)
        {
            a = p.getData();
        }
        void display()
        {
            cout << "Product = " << a << endl;
        }
```

```
};

int main()
{
Item i1;
Product p1(3,4);
i1=p1;
p1.showData();
i1.display();
return 0;
}
```

```
Output:
a = 3 b = 4
Product = 12
```

4. Create Product class and convert Product type to Item type using casting operator

```
int main()
{
Item i1;
Product p1;

p1.setData(3,4);
i1=p1;
return 0;
}
```

```
#include<iostream>
using namespace std;
class Item
{
    private:
        int i;
    public:
        Item(){}
        Item(int x)
        {
            i = x;
        }
        void display()
        {
            cout << "Product = " << i << endl;
        }
};
class Product
{
    private:
        int a, b;
```

```
    public:
        void setData(int x, int y)
        {
            a = x;
            b = y;
        }
        operator Item()
        {
            Item I(a*b);
            return I;
        }
        void display()
        {
            cout << "a = " << a << " b = " << b << endl;
        }
};

int main()
{
Item i1;
Product p1;
p1.setData(3,4);
i1=p1;
p1.display();
i1.display();
return 0;
}
```

```
Output:
a = 3 b = 4
Product = 12
```

5. Create two classes Invent1 and Invent2 and also add necessary constructors in it. Now add functions to support Invent1 to float and Invent1 to Invent2 type.

Example -

```
int main()
{
Invent1 s1(4,5);
Invent2 d1;
float tv;
tv=s1;
d1=s1;
return 0;
}
```

```
#include <iostream>
using namespace std;
```

```
class Invent2
{
private:
    float a, b;

public:
    Invent2() {}
    Invent2(float x, float y)
    {
        a = x;
        b = y;
    }
    void display()
    {
        cout << "Invent2" << endl;
        cout << "a = " << a << " b = " << b << endl;
    }
};
class Invent1
{
private:
    float x, y;

public:
    Invent1() {}
    Invent1(float a, float b)
    {
        x = a;
        y = b;
    }
    operator float()
    {
        return (x + y);
    }
    operator Invent2()
    {
        Invent2 i(x, y);
        return i;
    }
    void display()
    {
        cout << "Invent1" << endl;
        cout << "x = " << x << " y = " << y << endl;
    }
};

int main()
{
    Invent1 s1(4, 5);
    Invent2 d1;
    float tv;
    tv = s1; // Invent1 to float
    d1 = s1; // Invent1 to Invent2

    cout << "float = " << tv << endl;
    s1.display();
```

```
    d1.display();

    return 0;
}
```

```
Output:
float = 9
Invent1
x = 4 y = 5
Invent2
a = 4 b = 5
```

6. Create a Time class and take Duration in seconds. Now you need to convert seconds(i.e in int) to Time class.

Example-

```
int main()
{
int duration;
cout<<"Enter time duration in minutes";
cin>>duration;
Time t1 = duration;
t1.display();
return 0;
}
```

```
#include <iostream>
using namespace std;
class Time
{
private:
    int sec;

public:
    Time() {}
    Time(int x)
    {
        sec = x * 60;
    }
    void display()
    {
        cout << "Time duration is " << sec << " second" << endl;
    }
};
int main()
{
    int duration;
    cout << "Enter time duration in minutes: ";
    cin >> duration;
    Time t1 = duration;
    t1.display();
    return 0;
}
```

```
Output:
Enter time duration in minutes: 500
Time duration is 30000 second
```

7. Create two class Time and Minute and add required getter and setter including constructors.
Now you need to type cast Time object into Minute to fetch the minute from Time and display it.
Example -

```
int main()
{
Time t1(2,30);
t1.display();
Minute m1;
m1.display();
m1=t1 // Fetch minute from time
t1.display();
m1.display();
return 0;
}
```

```cpp
#include <iostream>
using namespace std;

class Minute
{
private:
    int m, s;

public:
    Minute()
    {
        m = 0;
        s = 0;
    }
    Minute(int x, int y)
    {
        m = x;
        s = y;
    }
    void display()
    {
        cout << "Minute class" << endl;
        cout << "Minute : " << m << " Second : " << s << endl;
    }
};

class Time
{
private:
    int m, s;

public:
    Time() {}
    Time(int x, int y)
    {
        m = x;
        s = y;
    }
    operator Minute()
    {
```

```
        Minute M(m, s);
        return M;
    }
    void display()
    {
        cout << "Time class" << endl;
        cout << "Minute : " << m << " Second : " << s << endl;
    }
};

int main()
{
    Time t1(2, 30);
    t1.display();
    Minute m1;
    m1.display();
    m1 = t1; // Fetch minute from time
    t1.display();
    m1.display();
    return 0;
}
```

```
Output:
Time class
Minute : 2 Second : 30
Minute class
Minute : 0 Second : 0
Time class
Minute : 2 Second : 30
Minute class
Minute : 2 Second : 30
```

8. Create a Rupee class and convert it into int. And Display it.

Example-

```
int main()

{
Rupee r = 10;
int x = r;
cout<<x;
return 0;
}
```

```
#include <iostream>
using namespace std;
class Rupee
{
private:
    int x;

public:
    Rupee() {}
    Rupee(int r)
    {
        x = r;
    }
    operator int()
```

```
    {
        return x;
    }
};
int main()
{
    Rupee r = 10;
    int x = r;
    cout << "x = " << x << endl;
    return 0;
}
```

```
==========================================================================
Output:
x = 10
```

9. Create a Dollar class and add necessary functions to support int to Dollar type conversion.
Example-

```
int main()
{
int x = 50;
Dollar d;
d = x;
d.display();
return 0;
}
```

```
#include <iostream>
using namespace std;
class Dollar
{
private:
    int x;

public:
    Dollar() {}
    Dollar(int d)
    {
        x = d;
    }
    void display()
    {
        cout << "Dollar : " << x << endl;
    }
};
int main()
{
    int x = 50;
    Dollar d;
    d = x;
    d.display();
    return 0;
}
```

```
==========================================================================
Output:
Dollar : 50
```

10. Create two classes Rupee and Dollar and add necessary functions to support Rupee to

Dollar and Dollar to Rupee conversion.
Example-

```
int main()
{
Rupee r = 23;
Dollar d = r; // Rupee to Dollar conversion
d.display();
r.display();
r = d; // Dollar to Rupee Conversion
d.display();
r.display();
return 0;
}
```

```cpp
#include <iostream>
using namespace std;

class Dollar
{
private:
    float x;

public:
    Dollar() {}
    float getD()
    {
        return x;
    }
    Dollar(float a)
    {
        x = a;
    }
    void display()
    {
        cout << "Dollar : " << x << endl;
    }
};

class Rupee
{
private:
    float x;

public:
    Rupee() {}

    Rupee(float a)
    {
        x = a;
    }
    Rupee(Dollar d)
    {
        x = (d.getD() * 82.17);
    }
    operator Dollar()
    {
        return (x / 82.17);
```

```
    }
    void display()
    {
        cout << "Rupee : " << x << endl;
    }
};


int main()
{
    Rupee r = 23;
    Dollar d = r; // Rupee to Dollar conversion
    d.display();
    r.display();
    r = d; // Dollar to Rupee Conversion
    d.display();
    r.display();
    return 0;
}
```

==========

```
Output:
Dollar : 0.279907
Rupee : 23
Dollar : 0.279907
Rupee : 23
```