LAB 4

1. /*Write a C++ program to illustrate the overloading of prefix increment operator.*/

```
#include<iostream>
using namespace std;
class counter
{
	private :
		int count;
	public:
		counter()
		{
			count=0;
		}
		int get_count()
		{
			return count;
		}
		void operator ++()
		{
			++count;
```

```
          }
};
int main()
{
    counter c1 , c2;
    cout<<"Value of count before increment "<<endl;
    cout<<"c1 count = "<<c1.get_count()<<endl;
    cout<<"c2 count = "<<c2.get_count()<<endl;
    ++c1;
    ++c2;
    ++c2;
    cout<<"Value of count after increment "<<endl;
    cout<<"c1 count = "<<c1.get_count()<<endl;
    cout<<"c2 count = "<<c2.get_count()<<endl;
}
```

2. /*Write a C++ program to illustrate the overloading of postfix increment operator.*/

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

```
class counter
{
        private :
                int count;
        public:
                counter()
                {
                        count=10;
                }
                int get_count()
                {
                        return count;
                }
                void operator ++(int)
                {
                        count++;
                }
};
int main()
{
        counter c1 , c2;
        cout<<"Value of count before increment "<<endl;
        cout<<"c1 count = "<<c1.get_count()<<endl;
        c1++;
        cout<<"Value of count after increment "<<endl;
        cout<<"c1 count = "<<c1.get_count()<<endl;
}
```

3. /*Write a C++ program to illustrate the overloading of prefix decrement operator.*/

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

class counter
{
        private :
                int count;
        public:
                counter()
                {
                        count=10;
                }
        int get_count()
        {
            return count;
        }
        void operator ++()
        {
            --count;
        }
};
int main()
{
    counter c1 ;

    cout<<"Value of count before decrement "<<endl;
    cout<<"c1 count = "<<c1.get_count()<<endl;
    ++c1;

    cout<<"Value of count after decrement "<<endl;
    cout<<"c1 count = "<<c1.get_count()<<endl;
}
```

4. /*Write a C++ program to illustrate the overloading of postfix decrement operator.*/

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

class counter
{
        private :
                int count;
        public:
                counter()
                {
                        count=10;
                }
                int get_count()
                {
                        return count;
                }
                void operator --(int)
                {
                        count--;
                }
};
int main()
{
        counter c1 ;

        cout<<"Value of count before decrement "<<endl;
        cout<<"c1 count = "<<c1.get_count()<<endl;
        c1--;

        cout<<"Value of count after decrement "<<endl;
        cout<<"c1 count = "<<c1.get_count()<<endl;
}
```

5. /*Write a program to compute subtraction of two complex numbers using operator overloading.*/

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

class complex
{
        private :
                int r , im;
        public:
                void getdata()
                {
                        cout<<"Enter the real and imaginary part of the
number :"<<endl;
                        cin>>r>>im;
                        cout<<endl;
                }
                complex operator -(complex c)
                {
                        complex c1;

                        c1.r = r-c.r;
                        c1.im = im - c.im;
                        return c1;
                }
                void display()
                {
                        cout<<endl<<"The sum of the imaginary
number is "<<r<<" + "<<im<<"i";
```

```
                }
};
int main()
{
        complex c1, c2, c3;
        c1.getdata();
        c2.getdata();
        c3 = c1-c2;

        c3.display();
}
```

6. /*Write a program to compute addition of two complex numbers using operator overloading.*/

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

class complex
{
        private :
                int r , im;
```

```
    public:
        void getdata()
        {
            cout<<"Enter the real and imaginary part of the
number :"<<endl;
            cin>>r>>im;
            cout<<endl;
        }
        complex operator +(complex c)
        {
            complex c1;

            c1.r = r+c.r;
            c1.im = im + c.im;
            return c1;
        }
        void display()
        {
            cout<<endl<<"The sum of the imaginary
number is "<<r<<" + "<<im<<"i";
        }
};
```

```
int main()
{
        complex c1, c2, c3;
        c1.getdata();
        c2.getdata();
        c3 = c1+c2;

        c3.display();
}
```

7. /*Write a C++ program to multiply two objects.*/

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

class num
{
        private :
                int a;
        public:
                num(int c)
                {
```

```
                a=c;
            }
            num operator *(num n)
            {
                num n1(0);

                n1.a = a * n.a;
            }
            void display()
            {
                cout<<"The product is "<<a<<endl;
            }
};
int main()
{
    num n1(15) , n2(10) , n3(0);

    n3 = n1*n2;

    n3.display();
}
```

8. /*Write a C++ program to illustrate modulo division of objects using operator overloading.*/

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

class mod
{
        private :
                int a;
        public:
                mod(int c)
                {
                        a=c;
                }
                mod operator %(mod n)
                {
                        mod n1(0);

                        n1.a = a % n.a;
                }
                void display()
        {
            cout<<"The modulo is "<<a<<endl;
        }
};
int main()
{
    mod n1(15) , n2(10) , n3(0);

    n3 = n1%n2;

    n3.display();
}
```

9. /*Write a C++ program to illustrate normal division of objects using operator overloading.*/

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

class div
{
    private :
```

```
                int a;
        public:
                div(int c)
                {
                        a=c;
                }
                div operator /(div n)
                {
                        div n1(0);

                        n1.a = a / n.a;
                }
                void display()
                {
                        cout<<"The quotient is "<<a<<endl;
                }
};
int main()
{
        div n1(15),n2(10),n3(0);

        n3 = n1/n2;
        n3.display();
}
```

10. /*Write a program that takes two amount (can be in RS and PS) as input and decide which one is less.*/

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

class Currency
{
        private:
                int rs , p;

        public:
                Currency(int a, int b)
                {
                        rs = a;
                        p = b;
                }
                void getdata()
```

```
		{
			cout<<"The currency is Rs. "<<rs<<" and 
"<<p<<" paisa"<<endl<<endl;
		}
		int operator <(Currency c)
		{
			p = rs*100+p;
			c.p = c.rs*100+c.p;

			if(p<c.p)
				return 1;
			else
				return 0;
		}
};
int main()
{
	Currency c1(10,50) , c2(50,60);

	c1.getdata();
	c2.getdata();
```

```
        if(c1<c2)
                cout<<"C1 is the lesser one";
        else
                cout<<"C2 is the lesser one";
}
```

11. /*Write a program to compare the two distances taken as input in the program and decide which one is greater than other.*/

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

class dist
{
        private:
                int x;

        public:
                dist(int a)
                {
                        x= a;
```

```
		}
		void getdata()
		{
			cout<<"The distance is "<<x<<endl<<endl;
		}
		int operator <(dist c)
		{
			if(x>c.x)
				return 1;
			else
				return 0;
		}
};
int main()
{
	dist c1(10) , c2(50);

	c1.getdata();
	c2.getdata();

	if(c1<c2)
		cout<<"C1 is the greater one";
	else
		cout<<"C2 is the greater one";
}
```

12. /*Write a C++ program to illustrate the overloading of relational
operator >.*/

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

class great
{
	int data;

	public:
		great(int d)
		{
			data = d;
		}
		int getdata()
		{
```

```
                return data;
            }
            int operator >(great d)
            {
                if(data>d.data)
                {
                    return 1;
                }
                else
                {
                    return 0;
                }
            }
};
int main()
{
    great d1(100) , d2(200);

    cout<<"Data present in objects "<<endl;
    cout<<"d1 data = "<<d1.getdata()<<endl;
    cout<<"d2 data = "<<d2.getdata()<<endl;
```

```
	if(d1>d2)
	{
		cout<<"d1 object is the greater one."<<endl;
	}
	else
	{
		cout<<"d2 object is the greater one."<<endl;
	}
	return 0;
}
```

13. /*Write a C++ program to illustrate the overloading of relational operator <.*/

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

class lesser
{
	int data;
```

```
    public:
        lesser(int d)
        {
            data = d;
        }
        int getdata()
        {
            return data;
        }
        int operator <(lesser d)
        {
            if(data<d.data)
            {
                return 1;
            }
            else
            {
                return 0;
            }
        }
};
```

```
int main()
{
        lesser d1(100) , d2(200);

        cout<<"Data present in objects "<<endl;
        cout<<"d1 data = "<<d1.getdata()<<endl;
        cout<<"d2 data = "<<d2.getdata()<<endl;

        if(d1<d2)
        {
                cout<<"d1 object is the lesser one."<<endl;
        }
        else
        {
                cout<<"d2 object is the lesser one."<<endl;
        }
        return 0;
}
```

14. /*Write a C++ program to illustrate the overloading of relational

operator <=.*/

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

class lessor_equal
{
	int data;

	public:
		lessor_equal(int d)
		{
			data = d;
		}
		int getdata()
		{
			return data;
		}
		int operator <=(lessor_equal d)
		{
			if(data<=d.data)
			{
				return 1;
```

```
                  }
                  else
                  {
                        return 0;
                  }
            }
};
int main()
{
      lessor_equal d1(100) , d2(100);

      cout<<"Data present in objects "<<endl;
      cout<<"d1 data = "<<d1.getdata()<<endl;
      cout<<"d2 data = "<<d2.getdata()<<endl;

      if(d1<=d2)
      {
            cout<<endl<<"d1 object is the less or equal to
d2."<<endl;
      }
      else
      {
```

```
		cout<<endl<<"d2 object is the less or equal to d1."<<endl;
	}
	return 0;
}
```

15. /*Write a C++ program to illustrate the overloading of relational
operator >=.*/

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

class greator_equal
{
	int data;

	public:
		greator_equal(int d)
		{
			data = d;
		}
```

```
        int getdata()
        {
            return data;
        }
        int operator >=(greator_equal d)
        {
            if(data>=d.data)
            {
                return 1;
            }
            else
            {
                return 0;
            }
        }
};
int main()
{
    greator_equal d1(100) , d2(200);

    cout<<"Data present in objects "<<endl;
    cout<<"d1 data = "<<d1.getdata()<<endl;
```

```
	cout<<"d2 data = "<<d2.getdata()<<endl;

	if(d1>=d2)
	{
		cout<<endl<<"d1 object is the greater or equal to 
d2."<<endl;
	}
	else
	{
		cout<<endl<<"d2 object is the greater or equal to 
d1."<<endl;
	}
	return 0;
}
```

```
16. /*Write a C++ program to illustrate the overloading of relational
operator ==.*/

#include<iostream>
using namespace std;

class is_equal
```

```
{
        int data;

        public:
                is_equal(int d)
                {
                        data = d;
                }
                int getdata()
                {
                        return data;
                }
                int operator ==(is_equal d)
                {
                        if(data==d.data)
                        {
                                return 1;
                        }
                        else
                        {
                                return 0;
                        }
                }
};
int main()
{
        is_equal d1(100) , d2(1000);

        cout<<"Data present in objects "<<endl;
        cout<<"d1 data = "<<d1.getdata()<<endl;
        cout<<"d2 data = "<<d2.getdata()<<endl;

        if(d1==d2)
        {
                cout<<endl<<"d1 object is equal to d2."<<endl;
        }
        else
        {
                cout<<endl<<"d1 object is not equal to d2."<<endl;
        }
        return 0;
}
```

17. /*Write a C++ program to illustrate the overloading of relational

operator !=.*/

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

class isnot_equal
{
	int data;

	public:
		isnot_equal(int d)
		{
			data = d;
		}
		int getdata()
		{
			return data;
		}
		int operator !=(isnot_equal d)
		{
```

```
                if(data!=d.data)
                {
                        return 1;
                }
                else
                {
                        return 0;
                }
        }
};
int main()
{
        isnot_equal d1(100) , d2(100);

        cout<<"Data present in objects "<<endl;
        cout<<"d1 data = "<<d1.getdata()<<endl;
        cout<<"d2 data = "<<d2.getdata()<<endl;

        if(d1!=d2)
        {
                cout<<endl<<"d1 object is not equal to d2."<<endl;
        }
        else
        {
                cout<<endl<<"d1 object is equal to d2."<<endl;
        }
        return 0;
}
```

18. /*Write a C++ program to illustrate the overloading of assignment operator.*/

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

class dist
{
        int feet , inch;

        public:
                dist(int f , int i)
                {
                        feet = f;
                        inch = i;
                }
                void operator =(dist &d)
                {
                        feet = d.feet;
                        inch = d.inch;
                }
                void display()
                {
                        cout<<"Feet   :   "<<feet<<"                    Inches   :
"<<inch<<endl;
                }
};
int main()
{
        dist d1(12,5) , d2(15,4);

        cout<<"First distance : "<<endl;
        d1.display();

        cout<<endl<<"Second distance : "<<endl;
        d2.display();
        d1=d2;

        cout<<endl<<"New first distance : "<<endl;
        d1.display();
}
```

19. /*Write a program to convert 10.5 kg weight into the object

of class Convert with data members kg and gram.*/

```
#include<iostream>
using namespace std;
class weight
{
        int kg , gram;
        public:
                weight(float m)
                {
                        kg = int(m);
                        gram = 1000*(m-kg);
                }
```

```
		void display()
		{
			cout<<"Kilogram : "<<kg<<endl<<"Gram : "<<gram<<endl;
		}
};
int main()
{
	float f=10.5;
	weight w = f;
	w.display();
	return 0;
}
```

20. /*Write a program to convert 100.25 rupee into the object of class Currency with data members RS and paisa.*/

```
#include<iostream>
using namespace std;
class currency
{
	int rs , paisa;
```

```
    public:
        currency(float m)
        {
            rs = int(m);
            paisa = 100*(m-rs);
        }
        void display()
        {
            cout<<"Rupees   :   "<<rs<<endl<<"Paisa   :
"<<paisa<<endl;
        }
};
int main()
{
    float f=100.25;
    currency c = f;
    c.display();
    return 0;
}
```

21. /*Write a program to convert kilogram into gram using user defined to user defined data conversion.(1kg = 1000gm).*/

```
#include<iostream>
using namespace std;
class kilogram
{
	int kg;
	public:
		kilogram(int d)
		{
			kg = d;
		}
		int getdata()
		{
			return kg;
		 }
};
class gram
{
	int gm;
```

```
        public:
                gram()
                {
                        gm = 0;
                }
                gram(kilogram kg)
                {
                        int g;
                        g = kg.getdata();

                        gm = g*1000;
                }
                void display()
                {
                        cout<<"Gram : "<<gm;
                }
};

int main()
{
        kilogram k(12);
```

```
    gram g = k;

    g.display();
}
```

22. /*Write a program to convert object from class that represents
temperature in Celsius scale to object of a class that represents it
in Fahrenheit scale.*/

```
#include<iostream>
using namespace std;
class celsius
{
    int c;
    public:
        celsius(int d)
        {
            c = d;
        }
        int getdata()
```

```
                {
                        return c;
                 }
};

class fahren
{
        int f;
        public:
                fahren()
                {
                        f = 0;
                }
                fahren(celsius cs)
                {
                        int c;
                        c = cs.getdata();
                        f = (c*9/5) + 32;
                }
                void display()
                {
                        cout<<"Fahrenheit : "<<f;
                }
};

int main()
{
        celsius c(100);
        fahren f = c;
        f.display();
}
```

23. /*Write a program to convert object from a class that represents
weight of gold in Nepal tola, to object of a class that represents
international gold measurement of weight in gram scale(1 tola =
11.664 gram) */

```
#include<iostream>
using namespace std;
class tola
{
        int t;
        public:
                tola(int d)
                {
                        t = d;
                }
                int getdata()
                {
                        return t;
                 }
};

class gram
{
        int g;

        public:
                gram()
                {
                        g = 0;
                }
                gram(tola to)
        {
            int gm;
            gm = to.getdata();
            g = gm*11.664;
        }
        void display()
        {
            cout<<"Gram : "<<g;
        }
};

int main()
{
    tola c(100);
    gram f = c;
    f.display();
}
```

24. /*Write a program to convert an object of dollar class to object of rupees class. Assume that dollar class has data members dol and cent an rupees class have data members rs and paisa.*/

```
#include<iostream>
using namespace std;
class dollar
{
        float d , c ;

        public:
                dollar(int a , int b)
                {
                        d=a;
                        c=b;
                }
                float getdata()
                {
                        float a;
                        a = d + c/100;
                        return a;
                }
};
class rupee
{
        int r , p;
```

```
	public:
		rupee()
		{
			r=0;
			p=0;
		}
		rupee(dollar e)
		{
			float x;
			x = e.getdata();
			x = x*108;
			r=int(x);
			p=(x-r)*100;
		}
		void display()
		{
			cout<<"The required dollar is "<<r<<" rupees
and "<<p<<" paisa";
		}
};
int main()
{
        dollar d(15.0,4);
        rupee r = d;

        r.display();
}
```