OOP LABORATORY 11

Name: **ANIRBAN HAZRA**
Section: **B-12**
Roll : **2005643**

1. Create a class which stores distance in feet and inches. Overload the ++ (post and pre) operator for the class for the statements D2=++D1 and D2=D1++.

Using Member function.
PROGRAM CODE:

```
#include<iostream>
using namespace std;
class Distance
{
   int f;
   int i;
public:

          void input()
          {
                  cin>>f>>i;
          }
   Distance operator++()
   {
      Distance t;

      cout<<"\nPre\n";
      if(i==11)
      {t.f=++f;
       t.i=0;
      }
      else{
      t.f=++f;
      t.i=++i;}
      return t;
   }
    Distance operator++(int)
   {
      Distance t;

      cout<<"\nPost\n";
      if(i==11)
      {t.f=f++;
```

```
        t.i=0;
        }
        else{
        t.f=f++;
        t.i=i++;}

        return t;
    }
    void dis()
    {

        cout<<"feet="<<f<<" inches="<<i<<endl;
    }
};
int main()
{
    Distance D1;
    cout<<"\nEnter the value of feets and inches= ";
    D1.input();
    cout<<"Original= \n";
    D1.dis();

    Distance D2;

    D2=++D1;
    D2.dis();
    D2=D1++;
    D2.dis();

}
```

OUTPUT:

```
Enter the value of feets and inches= 5 11
Original=
feet=5 inches=11

Pre
feet=6 inches=0

Post
feet=6 inches=0
```

Using Friend Function:

PROGRAM CODE:

```
#include<iostream>
using namespace std;
class Distance
{
   int f;
   int i;
public:

          void input()
          {
                    cin>>f>>i;
          }
   friend Distance operator++(Distance d)
   {
      Distance t;

      cout<<"Pre\n";
      if(d.i==11)
      {t.f=++d.f;
       t.i=0;
      }
      else{
      t.f=++d.f;
      t.i=++d.i;}
      return t;
   }
   friend Distance operator++(Distance d,int)
   {


      cout<<"Post\n";
      if(d.i==11)
      {d.f++;
       d.i=0;
      }
      else{
      d.f++;
      d.i++;}

      return d;
   }
   void dis()
   {

      cout<<"feet="<<f<<" inches="<<i<<endl;
   }
```

```
};
int main()
{
   Distance D1;
   cout<<"Enter the value of feets and inches= ";
   D1.input();
   D1.dis();
   cout<<"Original= \n";
   Distance D2;

   D2=++D1;
   D2.dis();
   D2=D1++;
   D2.dis();


}
```

OUTPUT:

```
Enter the value of feets and inches= 4 9
feet=4 inches=9
Original=
Pre
feet=5 inches=10
Post
feet=5 inches=10
```

2. Create a class which allocates the memory for a string through dynamic constructor. Overload

--the binary + to concatenate two strings and display it.

--relational operator < to compare the length of the two string.

Using member function

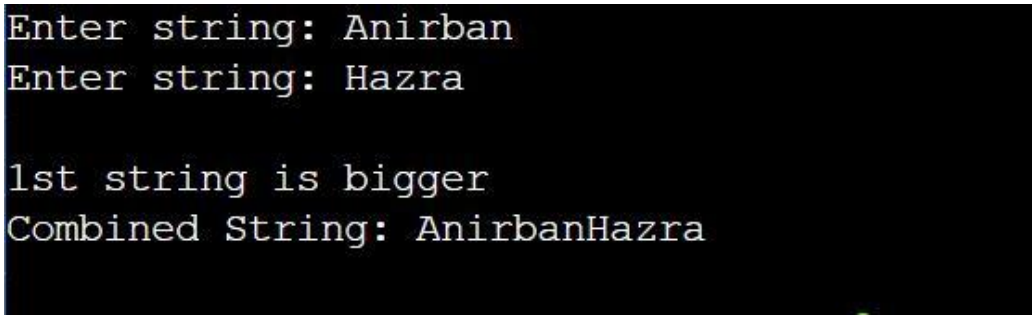
PROGRAM CODE:

```
#include <bits/stdc++.h>
using namespace std;


class String
{
   char* str;
public:
   String()
```

```
    {
       str = new char;

    }
    void input()
    {
       cout<<"Enter string: ";
       cin>>str;
    }
    void display()
    {
       cout<<"Combined String: "<<str;
    }
    String operator+(String s)
    {
       String obj;
       strcat(str,s.str);
       strcpy(obj.str,str);
       return obj;
    }
    int operator<(String s)
    {
       int a,b;
       a=strlen(str);
       b=strlen(s.str);

       if(a==b)
       {
          return 1;
       }else if(a>b)
       {
         return 2;
       }else
       {

          return 0;
       }


    }
};
int main()
{
    String str1,str2,str3;
    str1.input();
    str2.input();

    int x;

     if((str1<str2)==1)
```

```
        {
           cout<<"\nSame string"<<endl;
        }else if((str1<str2)==2)
        {
          cout<<"\n1st string is bigger"<<endl;
        }else
        {

           cout<<"\n2nd string is bigger"<<endl;
        }
        str3=str1+str2;
    str3.display();
    return 0;
}
```

OUTPUT:



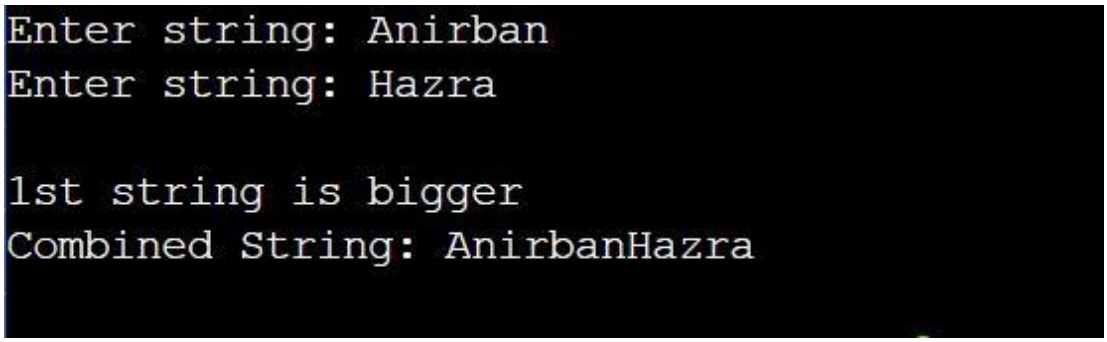
Using Friend Function

PROGRAM CODE:

```
#include <bits/stdc++.h>
using namespace std;


class String
{
    char* str;
public:
    String()
    {
        str = new char;

    }
    void input()
    {
        cout<<"Enter string: ";
        cin>>str;
    }
    void display()
    {
```

```
        cout<<"String: "<<str;
    }
    friend String operator+(String s,String s1)
    {
        String obj;
        strcat(s.str,s1.str);
        strcpy(obj.str,s.str);
        return obj;
    }
    friend int operator<(String s,String s1)
    {
        int a,b;
        a=strlen(s.str);
        b=strlen(s1.str);

        if(a==b)
        {
           return 1;
        }else if(a>b)
        {
          return 2;
        }else
        {

           return 0;
        }
    }
};
int main()
{
    String str1,str2,str3;
    str1.input();
    str2.input();

    int x;
if((str1<str2)==1)
        {
           cout<<"\nSame string"<<endl;
        }else if((str1<str2)==2)
        {
          cout<<"\n1st string is bigger"<<endl;
        }
else
        {
           cout<<"\n2nd string is bigger"<<endl;
        }
        str3=str1+str2;
    str3.display();
    return 0;
}
```

OUTPUT:



3. WAP to add two objects of time class. Overload the operator '==' to compare two objects and display whether they are equal or not.

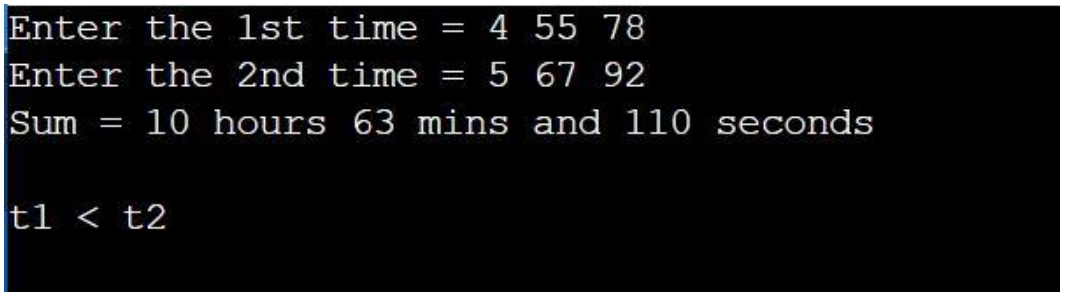
Using member function

PROGRAM CODE:

```
#include<iostream>

using namespace std;

class Time
{
        int h,m,s;
        public:
           Time()
           {

      h=0;
      m=0;
      s=0;
           }
                void input()
                {
                        cin>>h>>m>>s;
                }
                void dis()
                {
                        cout<<h<<" hours "<<m<<" mins and "<<s<<" seconds\n";
                }
                Time operator+(Time T2)
                {
                        Time T;
                        T.h=T2.h+h;
                        T.m=T2.m+m;
                        T.s=T2.s+s;
                        if(T.s>=60)
                        {
```

```
                        T.m++;
                        T.s-=60;
                }
                if(T.m>=60)
                {
                        T.h++;
                        T.m-=60;
                }
                return T;
        }
    int operator>(Time T2)
        {
                int s1,s2;
                s1=(h*3600)+(m*60)+s;
                s2=(T2.h*3600)+(T2.m*60)+T2.s;
                if(s1>s2)
                return 1;
                else
                return 0;
        }
};

int main()
{
        Time t1,t2,sum;
        cout<<"Enter the 1st time = ";
        t1.input();
        cout<<"Enter the 2nd time = ";
        t2.input();
        sum=t1+t2;
        cout<<"Sum = ";
        sum.dis();
        if(t1>t2)
                cout<<"\nt1 > t2";
        else
                cout<<"\nt1 < t2";
        return 0;
}
```

OUTPUT:



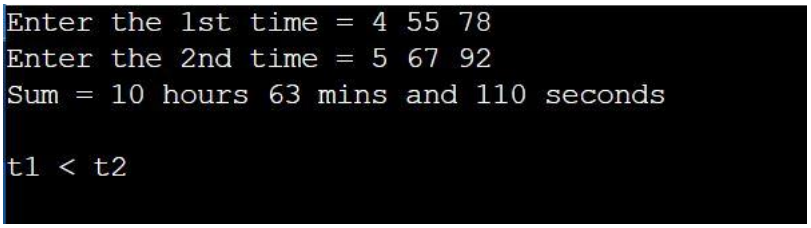
Using Friend Function

PROGRAM CODE:

```
#include<iostream>

using namespace std;

class Time
{
	int h,m,s;
	public:
		void input()
		{
			cin>>h>>m>>s;
		}
		void dis()
		{
			cout<<h<<" hours "<<m<<" mins and "<<s<<" seconds\n";
		}
		friend Time operator+(Time T1,Time T2)
		{
			Time T;
			T.h=T2.h+T1.h;
			T.m=T2.m+T1.m;
			T.s=T2.s+T1.s;
			if(T.s>=60)
			{
				T.m++;
				T.s-=60;
			}
			if(T.m>=60)
			{
				T.h++;
				T.m-=60;
			}
			return T;
		}
		friend int operator>(Time T1,Time T2)
		{
			int s1,s2;
			s1=(T1.h*3600)+(T1.m*60)+T1.s;
			s2=(T2.h*3600)+(T2.m*60)+T2.s;
			if(s1>s2)
			return 1;
			else
			return 0;
		}
};
```

```
int main()
{
        Time t1,t2,sum;
        cout<<"Enter the 1st time = ";
        t1.input();
        cout<<"Enter the 2nd time = ";
        t2.input();
        sum=t1+t2;
        cout<<"Sum = ";
        sum.dis();
        if(t1>t2)
                cout<<"\nt1 > t2";
        else if(t2>t1)
                cout<<"\nt1 < t2";
        else cout<<"\nt1 = t2";
        return 0;
}
```

OUTPUT:



4. WAP to overload unary operator for complex class. Overload the pre and post decrement operator to work for the statements
C1++;
++C1;
Overload any operator of your choice to find the modulus of a complex number.
[Note: void return type]

Using member function

PROGRAM CODE:

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

class complex
{
int a, b, c;
public:

complex()
{
}
```

```
void get_data()
{
cout << "Enter the Two Numbers:";
cin >> a>>b;
}

void operator++() //operater overloading function
{
a = ++a;
b = ++b;
}

void operator--() //operater overloading function
{
a = --a;
b = --b;
}

void display()
{
cout << a << "+\t" << b << "i" << endl;
}
};

main()
{
complex c;
c.get_data();
c++;
cout << "Increment Complex Number\n";
c.display();
c--;
cout << "Decrement Complex Number\n";
c.display();

}
```

OUTPUT:

```
Enter the two numbers: 3 5
Increment Complex Number
4 + 6i
Decrement Complex Number
2 + 4i
```

Using friend function
PROGRAM CODE:

```
class complex
{
int a, b, c;
public:
complex() {}
void get_data()
{
cout << "Enter the Two Numbers:";
cin >> a>>b;
}
friend void operator++(complex A) //operater overloading function
{
A.a = ++A.a;
A.b = ++A.b;
}
friend void operator--(complex A) //operater overloading function
{
A.a = --A.a;
A.b = --A.b;
}
void display()
{
cout << a << " + " << b << "i" << endl;
}
};

int main()
{
complex c;
c.get_data();
c++;
cout << "Increment Complex Number\n";
c.display();
c--;
cout << "Decrement Complex Number\n";
c.display();
}
```

OUTPUT:

```
Enter the two numbers: 3 5
Increment Complex Number
4 + 6i
Decrement Complex Number
2 + 4i
```