```
// 1. C++ program to demonstrate constructor overloading
#include <iostream>
using namespace std;

class numbers
{
  private:
     int n1,n2;

  public:
     // Constructor with no arguments
     numbers()
     {}

     // Constructor with arguments
     numbers(int m,int n)
     {
        n1=m;
        n2=n;
     }

     // Constructor with an argument
     numbers(int m)
     {
        n1=n2=m;
     }

     // copy constructor
     numbers(numbers& n)
     {
        n1=n.n1;
        n2=n.n2;
     }

     // to display the values of data members
     void showdata()
     {
        cout<<"\nn1= "<<n1;
        cout<<"\nn2= "<<n2;
     }
};

int main() {
   numbers obj1(10,20),obj2(30);

   cout << "\nObject1:";
   obj1.showdata();
   cout << "\n\nObject2:";
```

```
    obj2.showdata();
    numbers obj3(obj1),obj4=obj2;
    cout << "\n\nObject3:";
    obj3.showdata();
    cout << "\n\nObject4:";
    obj4.showdata();
    return 0;
}
```

```
// 2. C++ program to demonstrate constructor overloading - error
#include <iostream>
using namespace std;

class numbers
{
  private:
      int n1,n2;

  public:
      // Constructor with no arguments
      numbers()
      {}

      // Constructor with arguments
      numbers(int m,int n=0) //ambiguous - error
      {
         n1=m;
         n2=n;
      }

      // Constructor with an argument
      numbers(int m)
      {
         n1=n2=m;
      }

      // copy constructor
      numbers(numbers& n)
      {
         n1=n.n1;
         n2=n.n2;
      }

      // to display the values of data members
      void showdata()
      {
         cout<<"\nn1= "<<n1;
         cout<<"\nn2= "<<n2;
```

```
    }
};

int main() {
   numbers obj1(10,20),obj2(30);

   cout << "\nObject1:";
   obj1.showdata();
   cout << "\n\nObject2:";
   obj2.showdata();
   numbers obj3(obj1),obj4=obj2;
   cout << "\n\nObject3:";
   obj3.showdata();
   cout << "\n\nObject4:";
   obj4.showdata();
   return 0;
}
```

```
// C++ program to demonstrate constructor overloading; default arguments
#include <iostream>
using namespace std;

class numbers
{
  private:
     int n1,n2;

  public:
     // Constructor with no arguments
     numbers()
     {}

     // Constructor with default arguments
     numbers(int m,int n=0)
     {
        n1=m;
        n2=n;
     }

     // copy constructor
     numbers(numbers& n)
     {
        n1=n.n1;
        n2=n.n2;
     }

     // to display the values of data members
     void showdata()
```

```
    {
       cout<<"\nn1= "<<n1;
       cout<<"\nn2= "<<n2;
    }
};

int main() {
   numbers obj1(10,20),obj2(30);

   cout << "\nObject1:";
   obj1.showdata();
   cout << "\n\nObject2:";
   obj2.showdata();
   numbers obj3(obj1),obj4=obj2;
   cout << "\n\nObject3:";
   obj3.showdata();
   cout << "\n\nObject4:";
   obj4.showdata();
   return 0;
}
```

```
// C++ program to demonstrate constructor overloading - error
#include <iostream>
using namespace std;

class numbers
{
  private:
     int n1,n2;

  public:
     // Constructor with no arguments
     numbers()
     {}

     // Constructor with arguments
     numbers(int m,int n)
     {
        n1=m;
        n2=n;
     }

     // Constructor with default arguments
     numbers(int m,int n=0) //already exists - error
     {
        n1=m;
        n2=n;
```

```
    }

    // Constructor with an argument
    numbers(int m)
    {
       n1=n2=m;
    }

    // copy constructor
    numbers(numbers& n)
    {
       n1=n.n1;
       n2=n.n2;
    }

    // to display the values of data members
    void showdata()
    {
       cout<<"\nn1= "<<n1;
       cout<<"\nn2= "<<n2;
    }
};

int main() {
   numbers obj1(10,20),obj2(30);

   cout << "\nObject1:";
   obj1.showdata();
   cout << "\n\nObject2:";
   obj2.showdata();
   numbers obj3(obj1),obj4=obj2;
   cout << "\n\nObject3:";
   obj3.showdata();
   cout << "\n\nObject4:";
   obj4.showdata();
   return 0;
}
```

```
// C++ program to illustrate dynamic initialization
#include <iostream>
using namespace std;

class numbers
{
  private:
    int n1,n2;
```

```
  public:
     // Constructor with no arguments
     numbers()
     {}

     // Constructor with default arguments
     numbers(int m,int n=0)
     {
        n1=m;
        n2=n;
     }

     // copy constructor
     numbers(numbers& n)
     {
        n1=n.n1;
        n2=n.n2;
     }

     // to display the values of data members
     void showdata()
     {
        cout<<"\nn1= "<<n1;
        cout<<"\nn2= "<<n2;
     }
};

int main() {
   numbers obj,obj1(10,20),obj2(30);

   cout << "\nObject1:";
   obj1.showdata();
   cout << "\n\nObject2:";
   obj2.showdata();
   numbers obj3(obj1),obj4=obj2;
   cout << "\n\nObject3:";
   obj3.showdata();
   cout << "\n\nObject4:";
   obj4.showdata();
   cout<<"\n\nEnter 2 integers:";
   int x,y;
   cin>>x>>y;
   obj=numbers(x,y);
   cout << "\n\nObject:";
   obj.showdata();

   return 0;
}
```

Execution:

1. **Illustrate the use of constructors with complex class that has real and imaginary as members.**
2. **Illustrate the use of constructors with time class that has hours, minutes and seconds as members.**