```
/*_____________________________________________________________
_____________________
Code to understand the working of the copy constructor*/

#include <iostream>
using namespace std;

class Person{

private:
  string name;
  int age;

public:
  Person(string person_name, int person_age)
  {
      cout<<"Constructor for both name and age is
called"<<endl;
      name = person_name;
      age = person_age;
  }

  Person(const Person& obj)
  {
      cout<<"Copy constructor is called"<<endl;
      name = obj.name;
      age = obj.age;
  }
  void display()
  {
      cout<<"Name of current object: "<<name<<endl;
      cout<<"Age of current object: "<<age<<endl;
      cout<<endl;
```

```
  }

};
int main()
{
    Person obj1("First person",25);
    obj1.display();
    Person obj2(obj1);

    obj2.display();

    return 0;
}
/*_____
_____
Here is a program to overload two constructors, one to set
the name and age of a student with no parameters, and the
second to set the name and age of a student with a two-
parameter.
C++ program to demonstrate constructor overloading.*/

#include <iostream>

using namespace std;
class Student {
    private:
        string Name;
        int Age;
    public:
        Student() {
            Name = "Rohan";
            Age = 23;
        }
```

```
        Student(string str, int x) {
            Name = str;
            Age = x;
        }
        string get_Name() {
            return Name;
        }
        int get_Age() {
            return Age;
        }
};

int main() {
    Student stu1, stu2("Mohit", 25);
    cout << "Student1 Name: " << stu1.get_Name() << "
Student1 Age: " << stu1.get_Age() << endl;
    cout << "Student2 Name: " << stu2.get_Name() << "
Student2 Age: " << stu2.get_Age() << endl;
    return 0;
}

/*_____
_____
Here is a program to overload three constructors, one to
set the area with no parameters,
the second to set the area with one parameter, and the
third to set the area with two parameters.*/

#include <iostream>

using namespace std;
class Area {
    public:
```

```
        int area;
        Area() {
            area = 0;
        }
        Area(int side) {
            area = side * side;
        }
        Area(int length, int width) {
            area = length * width;
        }
        int disp() {
            return area;
        }
};

int main() {
    Area obj1;
    Area obj2(6);
    Area obj3(8, 5);
    cout << "Area of obj1: " << obj1.disp() << endl;
    cout << "Area of obj2: " << obj2.disp() << endl;
    cout << "Area of obj3: " << obj3.disp() << endl;
    return 0;
}

/*_____
_____
destructor */

#include <iostream>
using namespace std;

class Test{
```

```
  public :
  Test(){
    cout << "Test class object created.";
  }

  ~Test(){
    cout << endl << "Test class object destroyed.";
  }
};

int main() {
  Test testObj;
  return 0;
}

/*_____________________________________________________
Static Fields*/

#include <iostream>

using namespace std;

class Box {
   public:
      static int objectCount;

      Box(double l = 2.0, double b = 2.0, double h = 2.0)
{
         cout <<"Constructor called." << endl;
         length = l;
         breadth = b;
         height = h;
```

```
         objectCount++;
      }
      double Volume() {
         return length * breadth * height;
      }

   private:
      double length;
      double breadth;
      double height;
};

int Box::objectCount = 0;

int main(void) {
   Box Box1(3.3, 1.2, 1.5);
   Box Box2(8.5, 6.0, 2.0);

   cout << "Total objects: " << Box::objectCount << endl;

   return 0;
}

/*-----------------------------------------------*/

#include <iostream>
using namespace std;

class A {
public:
    static int year;
  static  int price  ;
  static  string model;
```

```
   A(int a,int b,string c)
   {year=a;
   price=b;
   model=c;}
   A()
   {year=2000;
   price= 100;
   model= "golf";}
  static void print() {

      cout << "A::staic method  is called\n";
     cout<<"yaer is " <<year<<"\t"<<"price is
"<<price<<"\t"<<
     " model is"<<model<<"\t"<<endl ;
  }

};
int  A::year=2000;
int   A:: price= 100;
string  A:: model= "golf";

int main() {

 A::print();
  A a(2022,500000,"BMW");

  a.print();
   A::print();
}

/*-----------------------------------------------*/

/#include <iostream>
```

```
using namespace std;

class Box {
   public:
      static int objectCount;

      Box(double l = 2.0, double b = 2.0, double h = 2.0)
{
         cout <<"Constructor called." << endl;
         length = l;
         breadth = b;
         height = h;

         objectCount++;
      }
      double Volume() {
         return length * breadth * height;
      }
      static int getCount() {
         return objectCount;
      }

   private:
      double length;
      double breadth;
      double height;
};

int Box::objectCount = 0;

int main(void) {
   cout << "Inital Stage Count: " << Box::getCount() <<
endl;
```

```
   Box Box1(3.3, 1.2, 1.5);
   Box Box2(8.5, 6.0, 2.0);

   cout << "Final Stage Count: " << Box::getCount() <<
endl;

   return 0;
}
/*-------------------------------------------------------
------------*/

#include<iostream>
using namespace std;

class Test
{
public:
  int x;
  int y;
public:
  Test(int x , int y ) { this->x = x; this->y = y; }
 Test( )
 { this->x = 26; this->y = 20; }

  void print() { cout << "x = " << x << " y = " << y <<
endl; }
};

int main()
{
  Test obj;
  obj.print();
```

```
  Test o (11 ,10);
  o.print();
  return 0;
}
/*-------------------------------------------------*/

#include<iostream>
using namespace std;
class Test
{
public:
  int x[30];

public:
  void input(int z)
{
for (int i =0;i<z;i++)
{
cout<<"enter element of array"<<endl;
cin>>x[i];
}
}

  void print()
  { for (int i =0;i<sizeof(x);i++)
     cout << x[i]<< endl; }
};

int main()
{
  Test obj;
  int y;
  cout<<"enter size of array"<<endl;
```

```
  cin>>y;
obj.input(y);
  obj.print();

  return 0;
}
```