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

template<class T1, class T2>
class Sample
{
	T1 a;
	T2 b;

	public:
		void get()
		{
			cout<<"Enter a and b: ";
			cin>>a>>b;
		}
		void disp()
		{
			cout<<"a="<<a<<"b="<<b<<endl;
		}

};

int main()
{
	Sample<int,int> s1;
	s1.get();
	s1.disp();
}
```

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

template<class T1, class T2>
class Sample
{
	T1 a;
	T2 b;

	public:
		void get()
		{
```

```
                cout<<"Enter a and b: ";
                cin>>a>>b;
        }
        void disp()
        {
                cout<<"a="<<a<<"b="<<b<<endl;
        }

};

int main()
{
        Sample<int,int> s1;
        s1.get();
        s1.disp();

        Sample<int,float> s2;
        s2.get();
        s2.disp();

        Sample<float, char> s3;
        s3.get();
        s3.disp();

        Sample<char, double> s4;
        s4.get();
        s4.disp();
}
```

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

template<class T>
class Student
{
        T marks1;
        T marks2;

        public:
                Student(T m1, T m2)
                {
```

```
                marks1=m1;
                marks2=m2;
        }
        T total_marks()
        {
                return (marks1+marks2);
        }
};

int main()
{
        Student<int> s1(50,60);
        Student<float> s2(56.5,70.5);

        cout<<s1.total_marks()<<endl;
        cout<<s2.total_marks();

}
```

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

template<class T>
class Student
{
        T marks1;
        T marks2;

        public:
                Student()
                {
                        marks1=10;
                        marks2=10;
                }
                Student(T m1, T m2)
                {
                        marks1=m1;
                        marks2=m2;
                }
                T total_marks()
                {
```

```
                return (marks1+marks2);
            }
};

int main()
{
    Student<int> s1(50,60);
    Student<float> s2(56.5,70.5);
    Student<int> s3;

    cout<<s1.total_marks()<<endl;
    cout<<s2.total_marks()<<endl;
    cout<<s3.total_marks();
}
```

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

template<class T>
class Array
{
    T *ptr;
    int size;

    public:
    Array(T[],int);
    void print();
};

template<class T>
Array<T>::Array(T arr[], int s)
{
    ptr=new T[s];
    size=s;

    for(int i=0;i<size;i++)
    ptr[i]=arr[i];
}

template<class T>
void Array<T>::print()
```

```
{
        for(int i=0;i<size;i++)
        cout<<*(ptr+i)<<"  ";
}
int main()
{
        int a[]={2,4,6,8,9};
        Array<int> ob(a,5);
        ob.print();
}
```

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

template<class T>
class XYZ
{
        static T s1;

        public:
                void put();
                static T s2;
};

template<class T>
T XYZ<T>::s1=1;

template<class T>
T XYZ<T>::s2=2;

template<class T>
void XYZ<T>::put()
{
        cout<<++s1<<"   ";
        cout<<++s2<<endl;
}

int main()
{
        XYZ<int> ob1;
        ob1.put();
```

```
	XYZ<int> ob2;
	ob2.put();

	XYZ<int> ob3;
	ob3.put();


}
```

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

template<class T>
class XYZ
{
	static T s1;

	public:
		void put();
		static T s2;
};

template<class T>
T XYZ<T>::s1=1;

template<class T>
T XYZ<T>::s2=2;

template<class T>
void XYZ<T>::put()
{
	cout<<++s1<<"   ";
	cout<<++s2<<endl;
}

int main()
{
	/*XYZ<int> ob1;
	ob1.put();
```

```
	XYZ<int> ob2;
	ob2.put();

	XYZ<int> ob3;
	ob3.put();
	*/
	XYZ<float> ob4;
	ob4.put();

	XYZ<float> ob5;
	ob5.put();

	XYZ<int> ob6;
	ob6.put();
}
```

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

template<class T>
class XYZ
{
	static T s1;

	public:
		void put();
		static T s2;
};

template<class T>
T XYZ<T>::s1=1;

template<class T>
T XYZ<T>::s2=2;

template<class T>
void XYZ<T>::put()
{
	cout<<++s1<<"   ";
	cout<<++s2<<endl;
}
```

```
int main()
{
	XYZ<int> ob1;
	ob1.put();

	XYZ<int> ob2;
	ob2.put();

	XYZ<int> ob3;
	ob3.put();

	XYZ<float> ob4;
	ob4.put();

	XYZ<float> ob5;
	ob5.put();

	XYZ<int> ob6;
	ob6.put();

	XYZ<float> ob7;
	ob7.put();
}
```

O/p:
2 3
3 4
4 5
2 3
3 4
5 6
4 5

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

template<class T, class U>
class A
{
```

```
 T x;
 U y;

 public:
 A()
 {
        cout<<"def constructor called: ";
 }
 A(T t, U u)
 {
        x=t;
        y=u;
        cout<<"param constructor called: ";
 }
 void disp()
 {
        cout<<"x="<<x<<"y="<<y<<endl;
 }
};

int main()
{
        A<int, char> ob1;
        ob1.disp();

        A<int, float> ob2(4,6.7);
        ob2.disp();
}
```

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

template<class T, class U=char>
class A
{

 T x;
 U y;

 public:
 A()
```

```
 {
        cout<<"def constructor called: ";
 }
 A(T t, U u)
 {
        x=t;
        y=u;
        cout<<"param constructor called: ";
 }
 void disp()
 {
        cout<<"x="<<x<<"y="<<y<<endl;
 }
};

int main()
{
        A<int, char> ob1;
        ob1.disp();

        A<int, float> ob2(4,6.7);
        ob2.disp();

        A<char> ob3('A','B');
        ob3.disp();

        A<int> ob4(12,66.4);
        ob4.disp();
}
```

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

template<class T=int, class U=char>
class A
{

 T x;
 U y;

 public:
```

```
 A()
 {
        cout<<"def constructor called: ";
 }
 A(T t, U u)
 {
        x=t;
        y=u;
        cout<<"param constructor called: ";
 }
 void disp()
 {
        cout<<"x="<<x<<"y="<<y<<endl;
 }
};

int main()
{
        A<int, char> ob1;
        ob1.disp();

        A<int, float> ob2(4,6.7);
        ob2.disp();

        A<char> ob3('A','B');
        ob3.disp();

        A<int> ob4(12,66.4);
        ob4.disp();

        A<> ob5(12,97);
        ob5.disp();

        A<> ob6(3.5, 65);
        ob6.disp();
}
```

O/p:

```
def constructor called: x=0y=
param constructor called: x=4y=6.7
param constructor called: x=Ay=B
param constructor called: x=12y=B
param constructor called: x=12y=a
```

param constructor called: x=3y=A

//Non-type parameters in class templates..

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

template<class T, int N>
class ABC
{
	T x;

	public:
		ABC(T a)
		{
			x=a;
		}
		void disp()
		{
			for(int i=1;i<=N;i++)
			cout<<x<<"  ";

			cout<<endl;
		}
};

int main()
{
	ABC<int,3> ob1(100);
	ob1.disp();

	ABC<float,4> ob2(200);
	ob2.disp();
}
```

//Supplying default values in non-type parameters of class templates..

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

```
template<class T, int N=5>
class ABC
{
	T x;

	public:
		ABC(T a)
		{
			x=a;
		}
		void disp()
		{
			for(int i=1;i<=N;i++)
			cout<<x<<"  ";

			cout<<endl;
		}
};

int main()
{
	ABC<int,3> ob1(100);
	ob1.disp();

	ABC<float> ob2(200);
	ob2.disp();

	ABC<char> ob3(65);
	ob3.disp();

	ABC<double,6> ob4(6.55);
	ob4.disp();
}
```

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

template<int N=5, class T=char>
class ABC
{
	T x;
```

```
	public:
		ABC(T a)
		{
			x=a;
		}
		void disp()
		{
			for(int i=1;i<=N;i++)
			cout<<x<<"  ";

			cout<<endl;
		}
};

int main()
{
	ABC<6> ob1('S');
	ob1.disp();

	ABC<> ob2(65);
	ob2.disp();

	ABC<3> ob3(97.6);
	ob3.disp();

}
```

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

template<int N, class T=char>
class ABC
{
	T x;

	public:
		ABC(T a)
		{
			x=a;
		}
```

```
                void disp()
                {
                        for(int i=1;i<=N;i++)
                        cout<<x<<"  ";

                        cout<<endl;
                }
};

int main()
{
        ABC<6> ob1('S');
        ob1.disp();

        ABC<5> ob2(65);
        ob2.disp();

        ABC<3> ob3(97.6);
        ob3.disp();

}
```

```
//Specialized function template..

#include<iostream>
using namespace std;

template<class T>
T square(T x)
{
        T res=x*x;
        return res;
}

template<>
string square<string> (string s)
{
        return (s+s);
}

int main()
```

```
{
	int i=3;
	int ii=square(i);
	cout<<ii<<"  ";

	double d=2.5;
	double dd=square(d);
	cout<<dd<<" ";

	char c='A';
	char cc=square(c);
	cout<<cc<<"  ";

	string s="A";
	string ss=square(s);
	cout<<ss;
}
```

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

template<class T>
T square(T x)
{
	T res=x*x;
	return res;
}

template<>
string square<string> (string s)
{
	return (s+s);
}

int main()
{
	int i=3;
	int ii=square<int>(i);
	cout<<ii<<"  ";

	double d=2.5;
```

```
	double dd=square<double>(d);
	cout<<dd<<" ";

	char c='A';
	char cc=square<char>(c);
	cout<<cc<<"  ";

	string s="A";
	string ss=square<string>(s);
	cout<<ss;
}
```

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

template<class T>
class Myclass
{
  T element;

  public:
  Myclass(T arg)
  {
	element=arg;
  }

  T increase()
  {
	return (++element);
  }
};

int main()
{
	Myclass<int> mi(5);
	Myclass<float> mf(5.5);
	Myclass<char> mc('A');

	cout<<mi.increase()<<" ";
	cout<<mf.increase()<<" ";
	cout<<mc.increase();
```

```
}
```

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

template<class T>
class Myclass
{
  T element;

  public:
  Myclass(T arg)
  {
        element=arg;
  }

  T increase()
  {
        return (++element);
  }
};

template<>
class Myclass<char>
{
        char element;

        public:
                Myclass(char arg)
                {
                        element=arg;
                 }
                char uppercase()
                {
                        if(element>='a' && element<='z')
                        return element-32;
                        else
                        return element;
                 }
};
int main()
```

```
{
        Myclass<int> mi(5);
        Myclass<float> mf(5.5);
        Myclass<char> mc('a');
        Myclass<char> mc2('B');

        cout<<mi.increase()<<" ";
        cout<<mf.increase()<<" ";
        cout<<mc.uppercase()<<" ";
        cout<<mc2.uppercase();
}
```

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

template<class T>
class Sample
{
   T elem;
   public:
   Sample(T e)
   {
      elem=e;
   }
   T incr();
   T convert();
};

template<class T>
T Sample<T>:: incr()
{
   elem++;
   return elem;
}

template <>
char Sample<char>::convert() //Instead of writing entire class template, just define the convert
function for char                                    //separately outside the class.
{
   elem=elem-32;
   return elem;
```

```
}
int main() {

    Sample<int> s1(2);
    cout<<s1.incr()<<endl;
    Sample<double> s2(2.5);
    cout<<s2.incr()<<endl;
    Sample<char> s3('a');
    cout<<s3.incr()<<endl;
    cout<<s3.convert();
}
```
