```
#include<iostream.h>
template<class T>
void swap(T x, T y)
{
   T temp;
   temp = x;
   x = y;
   y = temp;
   cout << x << y;
}
void main()
{
swap(7,8);
swap(3.4,5.4);
swap('a','b');
getch();
}
```

(2)

```
#include<iostream.h>
template<class T>
void show(T l, T m)
{
cout << l << m;
}
void main()
{
show(3,4);
show(1.5,1.6);
show('a','b');
getch();
}
```

class template for different data types.

```
template <class T, class U>
class Car
{
  T a;
  U b;
public:
  Car(T t1, U t2)
  {
    a = t1;
    b = t2;
  }
  void show()
  {
    cout << a << b;
  }
};

void main()
{
  Car <int, float> d1(4, 3.1);
  d1.show();
  Car <float, char> d2(3.4, 'a');
  d2.show();
  getch();
}
```


Q:- WAP to class template for sum of array elements

```
template <class T>
class Demo
{
  T a[5];
public:
  Demo()
  {
    int i;
    cout << "enter an array";
    for(i=0;i<=4;i++)
    {
      cin >> a[i];
    }
  }
  void show()
  {
    int i;
    T s=0;
```

```
for(i=0;i<5;i++)
{
  s=s+a[i];
}
cout<<s;
}
};
```

```
void main()
{
Demo<int> d1;
d1.show();
Demo<float> d2;
d2.show();
getch();
}
```

user defined types as template parameters

```
class A
{
public:
void show()
{
cout<<"Hello from A";
}
};
class B
{
public:
void show()
{
cout<<"Hello from B";
}
};
```

```
template<class T>
class Demo
{
T a;
public:
void showd()
{
a.show();
}
};
void main()
{
Demo<A> d1;
Demo<B> d2;
d1.showd();
d2.showd();
getch();
}
```

output:-

Hello from A

Hello from B

Exception Handling

errors can be divided into two categories:-

1. compile time errors
2. run time errors

Compile time errors are syntactic errors during the writing of the program. examples:- missing semicolon, missing comma, missing double quotes.

logical errors which are mainly due to improper understanding of the program logic by the programmer. Logical errors cause the unexpected or unwanted output.

Exceptions are runtime errors which a programmer usually does not expect. They occur accidentally which may result in abnormal termination of the program.

C++ provides exception handling mechanism which can be used to trap this exception and running programs smoothly after catching the exception.

examples:- divide by zero, violating array bounds

```
while(f1.eof()==0)
{
 f1.getline(str,15);
 cout<<str<<" ";
 f2.getline(str,15);
 cout<<str;
}
 f1.close();
 f2.close();
getch();
}
```

output :-

```
Kirti Jain
Rinki sharma
Charaya Goyal
```

Random Access in file :→ Random access means reading data randomly from any where in the file. For this purpose we need to set the position of the file pointer first in the file and then read the data.

(a) seekg() (b) seekp() (c) tellg() (d) tellp()

seekg and tellg are used when we are reading from file.

tellp and seekp are used when writing to the file.

the seek(p|g) function moves pointer to the specified position and tell(p|g) gives the position of the current pointer.