Queue:

```
public void exchange(int i , int j)
{
     Node <T> pi = head ;

     for(int k = 0 ; k < i ; k++)
          pi = pi.next;

     Node <T> pj = head;

     for ( int k = 0; k < j ; k ++)
          pj = pj . next ;

     pi.data = pj.data;
}

public void exchange ( int i , int j )
{
     T tmp = nodes[( head + i ) % maxsize];
     nodes[(head + i) % maxsize] = nodes[(head + j) % maxsize];
     nodes[(head + j) % maxsize] = tmp ;
}
```

Double Linked List

```
public void removeLast()
{
     if (head.next == null)
     {
          current = head = null ;
          return ;
     }

     DNode <T> p = head;

     while (p.next != null)
          p = p.next;

     p.previous.next = null;

     if (current == p)
          current = head ;
}
```

Linked List

```
public void exchange(int i,int j)
{
     //Go to ith node
     current = head;

     for (int k = 0 ; k < i ; k++)
          current = current.next;

     //Save ith node
     T temp1 = current.data;

     //Go to jth node
     current = head;

     for (int k = 0 ; k < j ; k++)
          current = current.next;

     //Save jth node
     T temp2 = current.data;

     //Replace jth node by ith node
     current.data = temp1;

     //Go to ith node
     current = head;

     for (int k = 0 ; k < i ; k++)
          current = current.next;

     //Replace ith node by jth node
     current.data = temp2;
}

public void findPrev()
{
     Node<T> p = head;

     while(p.next != current)
          p = p.next;

     current = current.next;
}
```

```
public void reverse()
{
     if (head != null && head.next != null)
     {
          Node<T> p = null;
          Node<T> cur = head;
          Node<T> q = head.next;

          while(cur != null)
          {
               cur.next = p;
               p = cur;
               cur = q;

               if (q != null)
                    q = q.next;
          }

          head = p;
     }
}
```