```
public void displayList()
{
     T x;

     if (! last())
     {
          x = retrieve();
          System.out.print(x + " ");
          findnext();
          displayList();
     }
     else
          System.out.print(retrieve() + " ");
}

public void displayListReverse()
{
     T x;

     if (! last())
     {
          x = retrieve();
          findnext();
          displayListReverse();
          System.out.print(x + " ");
     }
     else
          System.out.print(retrieve() + " ");
}
```

```
public void insertLast(T x)
{
    if (! last())
    {
        findnext();
        insertLast(x);
    }
    else
        insert(x);
}
```

```
public int countEvenData(ArrayList <Integer> l)
{
     int x;

     if (! last())
     {
          x = (Integer) retrieve();
          l.findnext();

          if (x % 2 == 0)
               return 1 + countEvenData(l);
          else
               return countEvenData(l);
     }
     else
     {
          x = (Integer) retrieve();
          if (x % 2 == 0)
               return 1;
          else
               return 0;
     }
}

public int sizeRec()
{
     if (! last())
     {
          findnext();
          return 1 + sizeRec();
     }
     else
          return 1;
}

public void copy(ArrayList <T> l)
{
     if (! last())
     {
          l.insert(retrieve());
          findnext();
          copy(l);
     }
     else
          l.insert(retrieve());
}
```

```
public void copyReverse(ArrayList <T> l)
{
     T x;

     if (! last())
     {
          x = retrieve();
          findnext();
          copyReverse(l);
          l.insert(x);
     }
     else
          l.insert(retrieve());
}



public int minimum()
{
     int min,y;

     if (! last())
     {
          y = (Integer) retrieve();
          findnext();
          min = minimum();

          if (y < min)
               return y;
          else
               return min;
     }
     else
          return (Integer) retrieve();
}
```

```
public static void copyStack(LinkStack<Integer> s1,LinkStack <Integer> s2)
{
     int x;

     if (! s1.empty())
     {
          x = s1.pop();
          copyStack(s1,s2);
          s1.push(x);
          s2.push(x);
     }
}

public static void displayLinkStackRec(LinkStack<Integer> s1)
{
     int x;

     if(! s1.empty())
     {
          x = s1.pop();
          System.out.print(x + " ");
          displayLinkStackRec(s1);
          s1.push(x);
     }
}

public static void displayLinkStackReverseRec(LinkStack<Integer> s1)
{
     int x;

     if(! s1.empty())
     {
          x = s1.pop();
          displayLinkStackReverseRec(s1);
          System.out.print(x + " ");
          s1.push(x);
     }
}
```

```
public static void reverseRec(LinkStack<Integer> s1,LinkStack<Integer> s2)
{
     int x;

     if(! s1.empty())
     {
          x = s1.pop();
          s2.push(x);
          reverseRec(s1,s2);
          s1.push(x);
     }
}

public static int sizeRecKeep(LinkStack<Integer> s1)
{
     int x,s;

     if(! s1.empty())
     {
          x = s1.pop();
          s = 1 + sizeRecKeep(s1);
          s1.push(x);
          return s;
     }
     else
          return 0;
}

public static int sizeRec(LinkStack<Integer> s1)
{
     int x;

     if(! s1.empty())
     {
          x = s1.pop();
          return 1 + sizeRec(s1);
     }
     else
          return 0;
}
```

```
public static void pushLast(LinkStack<Integer> s1,int newData)
{
      int x;
      if(! s1.empty())
      {
            x = s1.pop();
            pushLast(s1,newData);
            s1.push(x);
      }
      else
            s1.push(newData);
}

public static int countEvenData(LinkStack <Integer> s1)
{
      int x;
      if (! s1.empty())
      {
            x = s1.pop();

            if (x % 2 == 0)
                  return 1 + countEvenData(s1);
            else
                  return countEvenData(s1);
      }
      else
            return 0;
}

public static int countEvenDataKeep(LinkStack <Integer> s1)
{
      int x,s;
      if (! s1.empty())
      {
            x = s1.pop();

            if (x % 2 == 0)
                  s = 1 + countEvenDataKeep(s1);
            else
                  s = countEvenDataKeep(s1);

            s1.push(x);
            return s;
      }
      else
            return 0;
}
```

```
public static int minimum(LinkStack <Integer> s1)
{
     int min,y = 0;

     if (! s1.empty())
     {
          y = (Integer) s1.pop();
          min = minimum(s1);
          s1.push(y);

          if (y < min)
               return y;
          else
               return min;
     }
     else
          return Integer.MAX_VALUE;
}
```

```
public static void copyQueue(LinkQueue<Integer> q1,LinkQueue <Integer> q2)
{
     int x;

     if (q1.length() != 0)
     {
          x = q1.serve();
          q2.enqueue(x);
          copyQueue(q1,q2);
     }
}

public static void displayLinkQueueRec(LinkQueue<Integer> q1)
{
     int x;

     if(q1.length() != 0)
     {
          x = q1.serve();
          System.out.print(x + " ");
          displayLinkQueueRec(q1);
     }
}

public static void displayLinkQueueReverseRec(LinkQueue<Integer> q1)
{
     int x;

     if(q1.length() != 0)
     {
          x = q1.serve();
          displayLinkQueueReverseRec(q1);
          System.out.print(x + " ");
     }
}

public static void reverseRec(LinkQueue<Integer> q1,LinkQueue<Integer> q2)
{
     int x;

     if(q1.length() != 0)
     {
          x = q1.serve();
          reverseRec(q1,q2);
          q2.enqueue(x);
     }
}
```