# **Linked Lists**

1. **Generic list with insertBegin, insertEnd, insertMiddle, deletes, etc.**
2. **Count all the nodes in a list**
3. **Sum of all elements in a list**
4. **Search for a value in the list**
5. **Split a list into two**
6. **Merge two lists into one**
7. **Reverse a list**
8. **Sort list**
9. **Compare two lists and check is they are equal**

**class Node   
{  
 int data;  
 Node link;  
 Node()  
 {  
 data=0;  
 link=null;  
 }  
  
 Node(int data1, Node link1)  
 {  
 data=data1;  
 link=link1;  
 }  
}**

**import java.io.\*;  
class linklist  
{  
 static Node start;  
 linklist()  
 {  
 start=null;  
 }  
  
 void InsertB(int val)  
 {  
 Node nptr=new Node(val,null);  
 if(start==null)  
 {  
 start=nptr;  
 }  
 else  
 {  
 nptr.link=start;  
 start=nptr;  
 }  
 }  
  
 void InsertE(int val)  
 {  
 Node nptr=new Node(val,null);  
 Node temp=start;  
 while(temp.link!=null)  
 {  
 temp=temp.link;  
 }  
 temp.link=nptr;  
 }  
  
 void InsertM(int val,int p)  
 {  
 Node nptr=new Node(val,null);  
 Node temp=start;  
 int c=1;  
 while(temp.link!=null)  
 {  
 temp=temp.link;  
 c++;  
 if(c==p)  
 {  
 nptr.link=temp.link;  
 temp.link=nptr;  
 }  
 }  
 }  
  
 void Delete(int p)  
 {  
 Node temp=start;  
 int c=1;  
 while(temp.link!=null)  
 {  
 temp=temp.link;  
 c++;  
 if(c==p-1)  
 {  
 temp.link=temp.link.link;  
 }  
 }  
 }  
  
 int countNodes()  
 {  
 Node temp=start;  
 int c=1;  
 while(temp.link!=null)  
 {  
 temp=temp.link;  
 c++;  
 }  
 return c;  
 }  
  
 int sumNodes()  
 {  
 Node temp=start;  
 int c=0;  
 while(temp.link!=null)  
 {  
 c+=temp.data;  
 temp=temp.link;  
 }  
 c+=temp.data;  
 return c;  
 }  
  
 boolean search(int n)  
 {  
 Node temp=start;  
 boolean f=false;  
 while(temp.link!=null)  
 {  
 if(temp.data==n)  
 {  
 f= true;  
 break;  
 }  
 temp=temp.link;  
 }  
 if(temp.data==n)  
 {  
 f= true;  
 }  
 return f;  
 }  
  
 Node reverse(Node node)**

**{   
 Node prev = null;   
 Node current = node;   
 Node next = null;   
 while (current != null) {   
 next = current.link;   
 current.link = prev;   
 prev = current;   
 current = next;   
 }   
 node = prev;   
 return node;   
 }   
  
 void split (linklist l1)throws IOException   
 {  
 BufferedReader br=new BufferedReader(new InputStreamReader(System.in));  
 Node temp=start;  
 int n;  
 System.out.println("Enter node to be splitted at ");  
 n=Integer.parseInt(br.readLine());  
 int i;  
 for(i=1;i<n+1;i++)  
 {  
 temp=temp.link;  
  
 }  
 l1.start=temp.link;  
 temp.link=null;  
 System.out.println("First List");  
 Display();  
 System.out.println("Second List");  
 l1.Display();  
 }  
  
 void merge (linklist l1)  
 {  
 Node temp=this.start;  
 while(temp.link!=null)  
 {  
 temp=temp.link;  
 }  
 temp.link=l1.start;  
 }  
  
 void sort()  
 {  
 Node temp=start;  
 Node temp1=start;  
 Node temp2=start;  
 int c=0;  
 while (temp.link!=null)  
 {  
 temp=temp.link;  
 c=c+1;  
 }  
 c=c+1;  
 int i,k,t;  
 int a[]=new int [c];  
 for(i=0;i<c;i++)  
 {  
 a[i]=temp1.data;  
 temp1=temp1.link;  
 }  
 for (i=0;i<=c-2;i++)  
 {  
 for(k=0;k<=(c-2)-i;k++)  
 {  
 if(a[k]>a[k+1])  
 {  
 t=a[k];  
 a[k]=a[k+1];  
 a[k+1]=t;  
  
 }  
 }  
 }  
 for(i=0;i<c;i++)  
 {  
 temp2.data=a[i];  
 temp2=temp2.link;  
 }  
 }  
  
 void compare (linklist l1)  
 {  
 int c=0;  
 int c2=0;  
 boolean f1, f2;  
 f1=f2=false;  
 Node temp=this.start;  
 while (temp.link!=null)  
 {  
 c=c+1;  
 temp=temp.link;  
 }  
 c=c+1;  
 Node temp2=l1.start;  
 while(temp2.link!=null )  
 {  
 c2=c2+1;  
 temp2=temp2.link;  
 }  
 c2=c2+1;  
 if(c==c2)  
 f1=true;  
 if(f1==true)  
 System.out.println("1st condition satisfied ");  
 Node temp3=this.start;  
 Node temp4=l1.start;  
 int i,z;  
 z=0;  
 for (i=0;i<=c-2;i++)  
 {  
 if(temp3.data==temp4.data)  
 z=z+1;  
 temp3=temp3.link;  
 temp4=temp4.link;  
 }  
 if(temp3.data==temp4.data)  
 z=z+1;  
 if(z==c)  
 f2=true;  
 if(f1==true && f2==true)  
 System.out.println("The two lists are equal");  
 else  
 System.out.println("The two lists are not equal");  
 }  
  
 void Display()  
 {  
 Node temp=start;  
 while(temp.link!=null)  
 {  
 System.out.println(temp.data);  
 temp=temp.link;  
 }  
 System.out.println(temp.data);  
 }  
  
 public static void main(String args[])throws IOException  
 {  
 BufferedReader br=new BufferedReader(new InputStreamReader(System.in));  
 linklist l=new linklist();  
 linklist l1=new linklist();  
 int n,c;  
 while(true)  
 {  
 System.out.println("Menu");  
 System.out.println("1. Insert begin");  
 System.out.println("2. Insert end");  
 System.out.println("3. Insert anywhere");  
 System.out.println("4. Delete anywhere");  
 System.out.println("5. Count nodes");  
 System.out.println("6. Sum nodes");  
 System.out.println("7. Display");  
 System.out.println("8. Search");  
 System.out.println("9. Reverse");  
 System.out.println("10. Split");  
 System.out.println("11. Merge");  
 System.out.println("12. Sort");  
 System.out.println("13. Compare");  
 System.out.println("14. Exit");  
  
 System.out.println("Enter choice");  
 c=Integer.parseInt(br.readLine());  
 switch(c)  
 {  
 case 1:  
 {  
 System.out.println("Enter a number");  
 n=Integer.parseInt(br.readLine());  
 l.InsertB(n);  
 break;  
 }  
 case 2:  
 {  
 System.out.println("Enter a number");  
 n=Integer.parseInt(br.readLine());  
 l.InsertE(n);  
 break;  
 }  
 case 3:  
 {  
 System.out.println("Enter a number");  
 n=Integer.parseInt(br.readLine());  
 System.out.println("Enter position");  
 int p=Integer.parseInt(br.readLine());  
 l.InsertM(n,p);  
 break;  
 }  
 case 4:  
 {  
 System.out.println("Enter position");  
 int p=Integer.parseInt(br.readLine());  
 l.Delete(p);  
 break;  
 }  
 case 5:  
 {  
 System.out.println("Number of nodes is "+l.countNodes());  
 break;  
 }  
 case 6:  
 {  
 System.out.println("Sum of nodes is "+l.sumNodes());  
 break;  
 }  
 case 7:  
 {  
 l.Display();  
 break;  
 }  
 case 8:  
 {  
 System.out.println("Enter number you are searching for");  
 n=Integer.parseInt(br.readLine());  
 if(l.search(n))  
 {  
 System.out.println("Found");  
 }  
 else  
 {  
 System.out.println("These are not the numbers you are searching for");  
 }  
 break;  
 }  
 case 9:  
 {  
 System.out.println("Given Linked list");   
 l.Display();  
 start = l.reverse(start);   
 System.out.println("");   
 System.out.println("Reversed linked list ");   
 l.Display();  
 }  
 case 10:  
 {  
 l.split(l1);  
 break;  
 }  
 case 11:  
 {  
 l.merge(l1);  
 break;  
 }  
 case 12:  
 {  
 l.sort();  
 break;  
  
 }  
 case 13:  
 {  
 l.compare(l1);  
 break;  
 }  
 case 14:  
 {  
 System.exit(0);  
 }  
 }  
 }  
 }  
}**

**10. Implement stack using linked lists**

**import java.io.\*;  
class Stack  
{  
 private Node top;  
 public Stack() {  
 this.top = null;  
 }  
  
 public void push(int x)  
 {  
 Node node = new Node();  
 if (node == null)  
 {  
 System.out.print("Stack Overflow");  
 return;  
 }  
 System.out.println("Inserting " + x);  
 node.data = x;  
 node.link = top;  
 top = node;  
 }  
  
 public boolean isEmpty()  
 {  
 return top == null;  
 }  
  
 public int peek()  
 {  
 if (!isEmpty()) {  
 return top.data;  
 }  
 else {  
 System.out.println("Stack is empty");  
 return -1;  
 }  
 }  
  
 public void pop()  
 {  
 if (top == null)  
 {  
 System.out.print("\nStack Underflow");  
 return;  
 }  
 System.out.println("Removing " + peek());  
 top = (top).link;  
 }  
  
 void Display()  
 {  
 Node temp=top;  
 while(temp.link!=null)  
 {  
 System.out.println(temp.data);  
 temp=temp.link;  
 }  
 System.out.println(temp.data);  
 }  
  
 public static void main (String args[])throws IOException  
 {  
 BufferedReader br=new BufferedReader (new InputStreamReader (System.in));  
 int ch;  
 Stack S=new Stack();  
  
 while (true)  
 {  
 System.out.println("Enter choice ");  
 System.out.println("1.Push");  
 System.out.println("2.Pop");  
 System.out.println("3.Display");  
 System.out.println("4.Exit");  
 ch=Integer.parseInt(br.readLine());  
 switch(ch)  
 {  
 case 1:  
 System.out.println("Enter a no. ");  
 int n=Integer.parseInt(br.readLine());  
 S.push(n);  
 break;  
 case 2:  
 S.pop();  
  
 break;  
 case 3:  
 S.Display();  
 break;  
 case 4:  
 System.exit(0);  
 break;  
 }  
 }  
 }  
}**

**11. Implement queue using linked lists**

**import java.io.\*;  
class Queue   
{   
 static Node start;  
  
 Queue()   
 {   
 start=null;  
 }   
  
 void InsertB(int val)  
 {  
 Node nptr=new Node(val,null);  
 if(start==null)  
 {  
 start=nptr;  
 }  
 else  
 {  
 nptr.link=start;  
 start=nptr;  
 }  
 }  
  
 void InsertE(int val)  
 {  
 Node nptr=new Node(val,null);  
 Node temp=start;  
 while(temp.link!=null)  
 {  
 temp=temp.link;  
 }  
 temp.link=nptr;  
 }  
  
 boolean isEmpty()  
 {  
 return start == null;  
 }  
  
 int look()  
 {  
 if (!isEmpty()) {  
 return start.data;  
 }  
 else {  
 System.out.println("Queue is empty");  
 return -1;  
 }  
 }  
  
 void delete()   
 {  
 if (start == null)  
 {  
 System.out.print("\nQueue Underflow");  
 return;  
 }  
 System.out.println("Removing " + look());  
 start = (start).link;  
 }  
  
 void Display()  
 {  
 Node temp=start;  
 while(temp.link!=null)  
 {  
 System.out.println(temp.data);  
 temp=temp.link;  
 }  
 System.out.println(temp.data);  
 }  
  
 public static void main (String args[])throws IOException  
 {  
 BufferedReader br=new BufferedReader (new InputStreamReader (System.in));  
 int ch;  
 Queue Q=new Queue();  
  
 while (true)  
 {  
 System.out.println("Enter choice ");  
 System.out.println("1.Insert Front");  
 System.out.println("2.Insert Back");  
 System.out.println("3.Delete");  
 System.out.println("4.Display");  
 System.out.println("5.Exit");  
 ch=Integer.parseInt(br.readLine());  
 int n;  
 switch(ch)  
 {  
 case 1:  
 {  
 System.out.println("Enter a number");  
 n=Integer.parseInt(br.readLine());  
 Q.InsertB(n);  
 break;  
 }  
 case 2:  
 {  
 System.out.println("Enter a number");  
 n=Integer.parseInt(br.readLine());  
 Q.InsertE(n);  
 break;  
 }  
 case 3:  
 {  
 Q.delete();  
 break;  
 }  
 case 4:  
 {  
 Q.Display();  
 break;  
 }  
 case 5:  
 {  
 System.exit(0);  
 break;  
 }  
 }  
 }  
 }  
}**