Linked List:

Singly Linked List:

Creation:

```
public class create_sll{
   class Node{
      int data;
      Node next;
      public Node(int data){
         this.data=data;
         this.next=null;
      }
   }
   public Node head=null;
   public Node tail=null;
   public void addNode(int data){
      Node newNode=new Node(data);
      if(head==null){
         head=newNode;
         tail=newNode;
      }else{
         tail.next=newNode;
         tail=newNode;
      }
   }
   public void display(){
      Node curr=head;
      if(head==null){
         System.out.println("List is empty");
         return;
      }
      while(curr!=null){
         System.out.print(curr.data+" ");
         curr=curr.next;
      }
   }
   public static void main(String[] args){
      create_sll sll=new create_sll();
      sll.addNode(1);
      sll.addNode(2);
      sll.addNode(3);
      sll.addNode(4);
      sll.display();
   }
}
```

OUTPUT:

```
[Running] cd "c:\Data\Knowledge\DSA\Programs\ll\" && javac create_sll.java && java create_sll
1 2 3 4
```

Time complexity: O(n)

Insertion:

```
public class insert_sll{
   static class Node{
      int data;
      Node next;
      Node(int data){
         this.data=data;
         this.next=null;
      }
   }
   static Node insert_pos(Node head, int pos, int data){
      Node newNode=new Node(data);
      if(pos==1){
         newNode.next=head;
         head=newNode;
         return head;
      }
      Node curr=head;
      for(int i=1;i<pos-1 && curr!=null;i++){
         curr=curr.next;
      }
      if(curr==null){
         System.out.println("Position ut of bounds");
         return head;
      }
      newNode.next=curr.next;
      curr.next=newNode;
      return head;
   }
   static void printlist(Node head){
      while(head!=null){
         System.out.print(head.data+" ");
         head=head.next;
      }
   }
   public static void main(String[] args){
      Node head=new Node(3);
      head.next=new Node(5);
      head.next.next=new Node(8);
      head.next.next.next=new Node(10);
      int data=12, pos=2;
      head=insert_pos(head, pos, data);
      printlist(head);
   }
}
```

OUTPUT:

```
[Running] cd "c:\Data\Knowledge\DSA\Programs\ll\" && javac insert_sll.java && java insert_sll
3 12 5 8 10
```

Time complexity: O(n)

Deletion:

```
class Node{
   int data;
   Node next;
   Node(int data){
      this.data=data;
      this.next=null;
   }
}
public class delete_sll{
   public static Node delete_pos(Node head, int pos){
      Node temp=head;
      Node prev=null;
      if(temp==null) return head;
      if(pos==1){
         head=temp.next;
         return head;
      }
      for(int i=1;temp!=null && i<pos; i++){
         prev=temp;
         temp=temp.next;
      }
      if(temp!=null){
         prev.next=temp.next;
      }else{
         System.out.println("Position is not found");
      }   return head;
   }
   public static void printlist(Node head){
      while(head!=null){
         System.out.print(head.data+" ");
         head=head.next;
      }
   }
   public static void main(String[] args){
      Node head=new Node(1);
      head.next=new Node(2);
      head.next.next=new Node(3);
      head.next.next.next=new Node(4);
      head.next.next.next.next=new Node(5);
      int pos=2;
      head=delete_pos(head,pos);
      printlist(head);
   }
}
```

OUTPUT:

```
[Running] cd "c:\Data\Knowledge\DSA\Programs\ll\" && javac delete_sll.java && java delete_sll
1 3 4 5
```

Time complexity: O(n)

Deletion at middle:

```
class delete_mid_sll{
   static class Node{
      int data;
      Node next;
   }
   static Node delete_mid(Node head){
      if(head==null) return null;
      if(head.next==null) return null;
      Node slow=head;
      Node fast=head;
      Node prev=null;
      while(fast!=null && fast.next!=null){
         fast=fast.next.next;
         prev=slow;
         slow=slow.next;
      }
      prev.next=slow.next;
      return head;
   }
   static void printlist(Node ptr){
      while(ptr!=null){
         System.out.print(ptr.data+" ");
         ptr=ptr.next;
      }
   }
   static Node newNode(int data){
      Node temp=new Node();
      temp.data=data;
      temp.next=null;
      return temp;
   }
   public static void main(String[] args){
      Node head=newNode(1);
      head.next=newNode(2);
      head.next.next=newNode(3);
      head.next.next.next=newNode(4);
      head=delete_mid(head);
      printlist(head);
   }
}
```

OUTPUT:

```
[Running] cd "c:\Data\Knowledge\DSA\Programs\ll\" && javac delete_mid_sll.java && java delete_mid_sll
1 2 4
```

Time complexity: O(n)

Doubly Linked List:

Creation:

```
import java.util.*;
public class create_dll{
   class Node{
      int data;
      Node prev;
      Node next;
      public Node(int data){
         this.data=data;
      }
   }
   Node head, tail=null;
   public void addNode(int data){
      Node newNode=new Node(data);
      if(head==null){
         head=tail=newNode;
         head.prev=null;
         tail.next=null;
      }
      else{
         tail.next=newNode;
         newNode.prev=tail;
         tail=newNode;
         tail.next=null;
      }
   }
   public void display(){
      Node curr=head;
      if(head==null){
         System.out.println("List is empty");
         return;
      }
      while(curr!=null){
         System.out.print(curr.data+" ");
         curr=curr.next;
      }
   }
   public static void main(String[] args){
      create_dll dll=new create_dll();
      dll.addNode(1);
      dll.addNode(2);
      dll.addNode(3);
      dll.addNode(4);
      dll.addNode(5);
      dll.display();
   }
}
```

OUTPUT:

```
[Running] cd "c:\Data\Knowledge\DSA\Programs\ll\" && javac create_dll.java && java create_dll
1 2 3 4 5
```

Time complexity: O(n)

Insertion:

```
class Node{
   int data;
   Node prev;
   Node next;
   Node(int data){
      this.data=data;
   }
}
public class insert_dll{
   public static Node insert_pos(Node head, int pos, int data){
      Node newNode=new Node(data);
      if(pos==1){
         newNode.next=head;
         if(head!=null){
            head.prev=newNode;
         }
         head=newNode;
         return head;
      }
      Node curr=head;
      for(int i=1;i<pos-1 && curr!=null;i++){
         curr=curr.next;
      }
      if(curr==null){
         System.out.println("Position is out of bounds");
         return head;
      }
      newNode.prev=curr;
      newNode.next=curr.next;
      curr.next=newNode;
      if(newNode.next!=null){
         newNode.next.prev=newNode;
      }
      return head;
   }
   public static void printlist(Node head){
      Node curr=head;
      while(curr!=null){
         System.out.print(curr.data+" ");
         curr=curr.next;
      }
   }
   public static void main(String[] args){
      Node head=new Node(1);
      head.next=new Node(2);
      head.next.prev=head;
```

```
    head.next.next=new Node(4);
    head.next.next.prev=head.next;
    int data=3, pos=3;
    head=insert_pos(head, pos, data);
    printlist(head);
  }
}
```

OUTPUT:

```
[Running] cd "c:\Data\Knowledge\DSA\Programs\ll\" && javac insert_dll.java && java insert_dll
1 2 3 4
```

Time complexity: O(n)

Deletion:

```
class Node{
  int data;
  Node prev;
  Node next;
  Node(int data){
    this.data=data;
  }
}
public class delete_dll{
  static Node delete_pos(Node head, int pos){
    if(head==null) return head;
    Node curr=head;
    for(int i=1;curr!=null && i<pos;i++){
      curr=curr.next;
    }
    if(curr==null) return head;
    if(curr.prev!=null) curr.prev.next=curr.next;
    if(curr.next!=null) curr.next.prev=curr.prev;
    if(head==curr) head=curr.next;
    curr=null;
    return head;
  }
  static void printlist(Node head){
    Node curr=head;
    while(curr!=null){
      System.out.print(curr.data+" ");
      curr=curr.next;
    }
  }
  public static void main(String[] args){
    Node head=new Node(1);
    head.next=new Node(2);
    head.next.prev=head;
    head.next.next=new Node(3);
    head.next.next.prev=head.next;
    head=delete_pos(head, 2);
```

```
        printlist(head);
    }
}
```

OUTPUT:

```
[Running] cd "c:\Data\Knowledge\DSA\Programs\ll\" && javac delete_dll.java && java delete_dll
1 3
```

Time complexity: O(n)

Deletion at middle:

```
public class delete_mid_dll{
    class Node{
        int data;
        Node prev;
        Node next;
        public Node(int data){
            this.data=data;
        }
    }
    public int len=0;
    Node head, tail=null;
    public void addNode(int data){
        Node newNode=new Node(data);
        if(head==null){
            head=tail=newNode;
            head.prev=null;
            tail.next=null;
        }else{
            tail.next=newNode;
            newNode.prev=tail;
            tail=newNode;
            tail.next=null;
        }
        len++;
    }
    public void delete_mid(){
        if(head==null) return;
        else{
            Node curr=head;
            int mid=(len%2==0)?(len/2):((len+1)/2);
            for(int i=1;i<mid;i++){
                curr=curr.next;
            }
            if(curr==head){
                head=curr.next;
            }
            else if(curr==tail){
                tail=tail.prev;
            }else{
                curr.prev.next=curr.next;
```

```
            curr.next.prev=curr.prev;
        }
        curr=null;
    }
    len--;
  }
  public void display(){
    Node curr=head;
    if(head==null){
        System.out.println("List is empty");
        return;
    }
    while(curr!=null){
        System.out.print(curr.data+" ");
        curr=curr.next;
    }
  }
  public static void main(String[] args){
    delete_mid_dll dll=new delete_mid_dll();
    dll.addNode(1);
    dll.addNode(2);
    dll.addNode(3);
    dll.addNode(4);
    dll.addNode(5);
    dll.delete_mid();
    dll.display();
  }
}
```

OUTPUT:

```
[Running] cd "c:\Data\Knowledge\DSA\Programs\ll\" && javac delete_mid_dll.java && java delete_mid_dll
1 2 4 5
```

Time complexity: O(n)