① Remove At,   ② Add At.

```
public void removeAt(int idx){
    if(size==0){
        System.out.println("List is empty");
        return;
    }

    if(idx <0 || idx >= size){
        System.out.println("Invalid arguments");
        return;
    }

    if(idx ==0) removeFirst();
    else if( idx== size-1) removeLast();
    else{
        Node prev = head;
        for(int i=0; i< idx; i++)
            prev = prev.next;
        prev.next= prev.next.next;
        size--;
    }
}


public void addAt (int idx , int val){
    if(idx<0 || idx > size){
        System.out.println("Invalid arguments");
        return;
    }

    if( idx ==0) addFirst (val);
    else if(idx == size) addLast(val);
    else {
        Node temp = new Node();
        temp.data = val;
        Node prev = head;
        for(int i=0; i< idx-1 ; i++)
            prev= prev.next;

        temp.next = prev.next;
        prev.next = temp;
        size++;
    }
}
```

② Linked list Recursive.

a) Data Iterative
b) Pointer Iterative
c) Data Recursive
d) Pointer recursive

ⓐ Data Iterative (same as in arrays)

```
left= 0, right= size-1;
while(left <right){
    Node leftNode = getNodeAt(left);
    Node rightNode = getNodeAt (right);
    swapData( leftNode, rightNode);
    left++, right--;
}
```

(b)

```
Node prev = null, curr = head;          ⟵ Pointer Iterative
while (curr != null) {
    Node ahead = curr.next;
    curr.next = prev;
    prev = curr;
    curr = ahead;
}
swap(head, tail);
```

(c) Data recursive : CODE :

```
public Node reverseDR( Node left , Node right) {
    if( left == null)
        return right;

    right = reverseDR (left.next , right, counter+1);

    if(counter < size /2)
        swap (left, right);

    right = right.next ;
    return right;
}
static Node right;
public void reverseDR() {
    Node left = head;
    right = head;
     reverseDR(left, right);
}
```

d) Pointer recursive : CODE:

```
public void reversePRhelper (Node node){
    if(node== null || node.next== null)
        return;
    reversePRhelper(node.next);
    node.next.next = node;
}


public void reversePR(){
    Node curr= head;
    reversePRhelper (curr);

    Node temp= head;
    head = tail;
    tail = temp;

    tail.next = null;
}
```