DAY 1

import java.util.Scanner;

class Node

{

int data;

Node next;

Node prev;

Node(int data)

{

this.data=data;

this.next=null;

this.prev=null;

}

}

class doubleLinked {

Node head;//null value

Node tail;

//to add value

void add(int data)

{

Node newnode=new Node(data);

if(head==null)

{

head=newnode;

}

else{

Node temp=head;

while(temp.next!=null){

temp=temp.next;

}

temp.next=newnode;

newnode.prev=temp;

tail=newnode;

}

}

//printforward

void printforward()

{

Node temp=head;  
 System.out.println("forward traversal");  
while(temp!=null)  
{  
 //   
System.out.print(temp.data + "-->");  
  
temp=temp.next;  
}  
 System.out.println();  
}  
//print backward  
void printbackward()  
{  
Node temp=tail;  
System.out.println("backward traversal");  
 while(temp!=null){  
//   
System.out.print(temp.data + "-->");  
 temp=temp.prev;  
}  
}  
//to find middle value and insert value  
void middle(int val){  
 Node slow=head;  
 Node du=head;  
 Node fast=head;  
 Node newnode=new Node(val);  
   
 while(fast!=null&&fast.next!=null){  
 fast=fast.next.next;  
 slow=slow.next;  
   
 }  
 Node temp=slow.prev;  
 temp.next=newnode;  
 newnode.prev=temp;  
 newnode.next=slow;  
 slow.prev=newnode;  
 System.out.println("\n"+" element is inserted at Middle in the list");  
 while(du!=null){  
 System.out.print(du.data + "-->");  
du=du.next;  
 }  
   
}  
  
//insert at begining  
  
void begin(int val) {  
Node temp = head;  
Node newnode = new Node(val);  
head = newnode;  
if (temp != null) {  
 temp.prev = newnode;  
 newnode.next = temp;  
}  
newnode.prev = null;  
  
System.out.println("\nInsert at start");  
Node traverse = head;  
while (traverse != null) {  
 System.out.print(traverse.data + "-->");  
 traverse = traverse.next;  
}  
System.out.println();

}

//insert at ending  
void last(int val){  
   
 Node newnode=new Node(val);  
 if (head == null) {  
 head = newnode;  
} else {  
 Node temp = head;  
 while (temp.next != null) {  
 temp = temp.next;  
 }  
  
 temp.next=newnode;  
 newnode.prev=temp;  
}  
 Node temp=head;  
   
 while(temp!=null){  
 System.out.print(temp.data + "-->");  
temp=temp.next;  
}  
 System.out.print("\n");  
}  
//delete fist,last,middle elements

void deleteMiddle() { if (head == null) return;

Node slow = head;  
Node fast = head;  
  
// Find middle node  
while (fast != null && fast.next != null) {  
 fast = fast.next.next;  
 slow = slow.next;  
}  
  
// If middle is head  
if (slow.prev == null) {  
 head = slow.next;  
 if (head != null) head.prev = null;  
} else {  
 slow.prev.next = slow.next;  
}  
  
// If middle is tail  
if (slow.next == null) {  
 tail = slow.prev;  
 if (tail != null) tail.next = null;  
} else {  
 slow.next.prev = slow.prev;  
}  
  
slow.next = null;  
slow.prev = null;

} void deleteHead() { if (head == null) return; // List empty, nothing to delete

Node temp = head;  
head = head.next;  
  
if (head != null) {  
 head.prev = null;  
} else {  
 // List became empty after deleting head  
 tail = null;  
}  
  
temp.next = null; // Help GC

}

void deleteTail() { if (tail == null) return; // List empty, nothing to delete

if (tail.prev == null) {  
 // Only one node in the list  
 head = null;  
 tail = null;  
} else {  
 Node oldTail = tail;  
 tail = tail.prev;  
 tail.next = null;  
 oldTail.prev = null;   
}

}

}

public class Main

{

public static void main(String[] args) {

doubleLinked list=new doubleLinked();

Scanner sc=new Scanner(System.in);

int n=sc.nextInt();

while(n>0){

int data=sc.nextInt();

list.add(data);

n--;

}

list.printforward();

list.printbackward();

list.middle(70);

int val=60;

list.begin(val);

list.last(val);

list.deleteHead();

list.deleteTail();

list.deleteMiddle();

list.printforward();

}

}