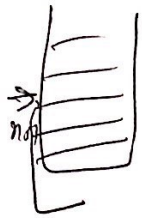
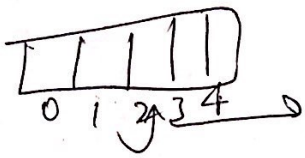
```
//Program to perform insertion and deletion in a doubly linked list
class DoublyList{
        DoublyNode head;
        void insertAtBeg(int item){
                DoublyNode newNode = new DoublyNode();
                newNode.data = item;
                newNode.next = head;
                newNode.prev = null;
                if(head == null)
                        head = newNode;
                else{

                        head.prev = newNode;
                        head = newNode;
                }
        }
        void deleteAtBeg(){
                DoublyNode temp = head;
                head = head.next;
                head.prev = null;
                temp.next = null;
        }
        void printList(){
                DoublyNode temp = head;
                while(temp != null){
                        System.out.print("<- "+temp.data+" ->");
                        temp = temp.next;
                }
        }

        public static void main(String[] args){
                DoublyList list = new DoublyList();
                list.insertAtBeg(10);
                list.insertAtBeg(20);
                list.insertAtBeg(30);
                list.insertAtBeg(40);
                list.insertAtBeg(50);
                list.insertAtBeg(60);
                System.out.println("Doubly Linked List is: ");
                list.printList();
                list.deleteAtBeg();
                System.out.println();
                System.out.println("List after deletion from beginning: ");
                list.printList();
        }
}
```

DoublyNode → Arrays

```
class DoublyNode {
    int data
    DoublyNode next;
    DoublyNode Prev;
    DoublyNode ( )
      { data=0
        next = null;
        prev = null;
      }
```

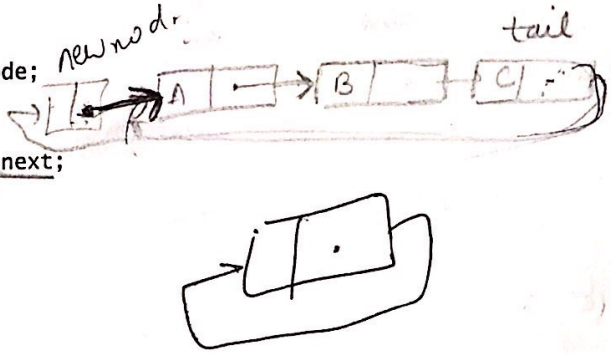




```
//template for Doubly Node                    DoublyNode
class DoublyNode{
        int data;
        DoublyNode next;
        DoublyNode prev;
        DoublyNode(){
                data = 0;
                next = null;
                prev = null;
        }
}
```

CircularList

```
//Program to perform insertion and deletion at the beginning of a circular linked
list
class CircularList{
        Node tail;
        void insertAtBeg(int item){
                Node newNode = new Node();
                newNode.data = item;
                if(tail == null){
                        newNode.next = newNode;
                        tail = newNode;
                }
                else{
                        newNode.next = tail.next;
                        tail.next = newNode;
                }
        }
        void deleteAtBeg(){
                Node temp = tail.next;
                tail.next = temp.next;
                temp.next = null;

        }
        void printList(){
                Node temp = tail.next;
                while(temp != tail){
                        System.out.print("-> "+temp.data);
                        temp = temp.next;
                }
                System.out.print("-> "+temp.data);
        }

        public static void main(String [] args){
                CircularList list = new CircularList();
                list.insertAtBeg(10);
                list.insertAtBeg(20);
                list.insertAtBeg(30);
                list.insertAtBeg(40);
                list.insertAtBeg(50);
                System.out.println("My Circular List is: ");
                list.printList();
                System.out.println();
                System.out.println("List after deletion from the beginning: ");
                list.deleteAtBeg();
                System.out.println("Updated list is: ");
                list.printList();
        }
}
```



StackByArray

```
package stack;

public class StackByArray {

	int[] arr;
	int topofStack;//keeps track of the cell which is last occupied in Array,
this will help in insertion/deletion

	public StackByArray(int size) {
		this.arr = new int[size];
		this.topofStack = -1;
		System.out.println("Successfully created an empty Stack of Size:
"+size);
	}//end of method

	public void push(int value) {
		//if array is full, show stack overflow error
		if (isFullStack()) {
			System.out.println("Stack overflow error!!");
		} else {
			arr[topofStack+1] = value;
			topofStack++;
			System.out.println("Successfully inserted " + value + " in
the stack");
		}
	}//end of method

	public void pop() {
		//if array is empty, show stack underflow error
		if (isEmptyStack()) {
			System.out.println("Stack underflow error!!");
		} else {
			System.out.println("Poping value from Stack: " +
arr[topofStack] + "...");
			topofStack--;
		}
	}//end of method

	public boolean isEmptyStack() {
		//if top pointer is zero, the stack is empty
		if (topofStack == -1)
			return true;
		else
			return false;
	}//end of method

	public boolean isFullStack() {
		if (topofStack == arr.length-1) {
			System.out.println("Stack is full !");
			return true;
		}else {
			return false;
		}
	}//end of method

	public void peekOperation() {
		if (!isEmptyStack())
			System.out.println("Top of Stack: " + arr[topofStack]);
		else {
			System.out.println("The stack is empty!!");
		}
		System.out.println();System.out.println();
	}//end of method

	public void deleteStack() {
		arr = null;
		System.out.println("Stack is successfully deleted");
	}//end of method

}//end of class
```

StackByArray (3)

```
package stack;

public class StackByArray {

        int[] arr;
        int topOfStack;//keeps track of the cell which is last occupied in Array,
  this will help in insertion/deletion


        public StackByArray(int size) {
                this.arr = new int[size];
                this.topOfStack = -1;
                System.out.println("Successfully created an empty Stack of Size:
   "+size);
        }//end of method


        public void push(int value) {
                //if array is full, show stack overflow error
                if (isFullStack()) {
                        System.out.println("Stack overflow error!!");
                } else {
                        arr[topOfStack+1] = value;
                        topOfStack++;
                        System.out.println("Successfully inserted " + value + " in
    the stack");
                }
        }//end of method


        public void pop() {
                //if array is empty, show stack underflow error
                if (isEmptyStack()) {
                        System.out.println("Stack underflow error!!");
                } else {
                        System.out.println("Poping value from Stack: " +
  arr[topOfStack] + "...");
                        topOfStack--;
                }
        }//end of method


        public boolean isEmptyStack() {
                //if top pointer is zero, the stack is empty
                if (topOfStack == -1)
                        return true;
                else
                        return false;
        }//end of method


        public boolean isFullStack() {
                if (topOfStack == arr.length-1) {
                        System.out.println("Stack is full !");
                        return true;
                }else {
                        return false;
                }
        }//end of method


        public void peekOperation() {
                if (!isEmptyStack())
                        System.out.println("Top of Stack: " + arr[topOfStack]);
                else {
                        System.out.println("The stack is empty!!");
                }
                System.out.println();System.out.println();
        }//end of method


        public void deleteStack() {
                arr = null;
                System.out.println("Stack is successfully deleted");
        }//end of method

}//end of class
```

StackByArrayMain

```
package stack;

public class StackByArrayMain {
	public static void main(String[] args) {


		System.out.println("Creating a stack of size 5...");
		StackByArray stack = new StackByArray(5);


		System.out.println("Pushing 6 values in the Stack...");
		for(int i=0; i<=5; i++) {
			stack.push(i*10);
		}


		System.out.println("Peeking value from stack");
		stack.peekOperation();


		System.out.println("Poping 6 values from the Stack...");
		for(int i=0; i<=5; i++) {
			stack.pop();
		}


		System.out.println("Deleting the Stack...");
		stack.deleteStack();

	}
}
```