**STACK**

**//lesson1**

**import java.util.\*;**

**public class Stack{**

**private int max;**

**private int top;**

**private int[] stack;**

**public Stack(int m){ //constructor**

**max=m;**

**top=-1;**

**stack= new int[max];**

**}**

**public void push(int num){**

**if (top==max-1){**

**System.out.println("Stack is full");**

**}**

**else {**

**top++;**

**stack[top]=num;**

**System.out.println(num+"success insert");**

**}**

**}**

**public void pop(){**

**if (top==-1){**

**System.out.println("Stack is empty");**

**}**

**else {**

**System.out.println(stack[top]+"delete success ");**

**top--;**

**}**

**}**

**public void display(){**

**if(top==-1)**

**System.out.println("stack empty");**

**else if(top==max-1)**

**System.out.println("stack full");**

**else{**

**System.out.println("values in stacks are");**

**for(int i=top;i>=0;i--){**

**System.out.println(stack[i]+"\t");**

**}**

**}**

**}**

**public static void main(String args[]){**

**Scanner in=new Scanner(System.in);**

**System.out.println("enter size of stack;");**

**int size=in.nextInt();**

**Stack s=new Stack(size);**

**char ch;**

**do{**

**System.out.println("menu");**

**System.out.println("1.push");**

**System.out.println("2.pop");**

**System.out.println("3.display");**

**System.out.println("4.exit");**

**System.out.print("\nEnter your choice;");**

**int choice=in.nextInt();**

**switch(choice){**

**case 1:**

**System.out.println("\nEnter number;");**

**int num=in.nextInt();**

**s.push(num);**

**break;**

**case 2:**

**s.pop();**

**break;**

**case 3:**

**s.display();**

**break;**

**case 4:**

**System.exit(0);**

**break;**

**}**

**System.out.println("continue(Y/N)");**

**ch=in.next().charAt(0);**

**}while(ch=='Y' || ch=='y');**

**}**

**}**

**arrary**

**import java.util.Scanner;**

**class AllArray{**

**public static void main(String args[]){**

**Scanner in = new Scanner (System.in);**

**System.out.print("Enter the size of the array : ");**

**int n = in.nextInt();**

**int arr[] = new int[n];**

**System.out.println();**

**//Insertion**

**for(int i = 0; i < n ; i++){**

**System.out.print("Enter the value in arr["+i+"]: ");**

**arr[i]=in.nextInt();**

**System.out.println();**

**}**

**//Displaying**

**for(int j =0; j <n; j++){**

**System.out.print("|in"+j+" |");**

**}**

**System.out.println();**

**for(int i = 0; i<n; i++){**

**System.out.print("| "+arr[i]+" |");**

**}**

**System.out.println("\n");**

**//Searching**

**int s,flag=0;**

**System.out.print("Enter a number to search: ");**

**s=in.nextInt();**

**for(int i = 0 ; i < n ; i++){**

**if(s==arr[i]){**

**System.out.println(">>"+arr[i]+"<< found at Index :: a["+i+"]");**

**flag=1;**

**}**

**}**

**if(flag==0){**

**System.out.println("Index not found");**

**}**

**System.out.println();**

**//Update**

**int f=0,u;**

**System.out.print("Which number do you want to update: ");**

**u = in.nextInt();**

**System.out.println();**

**for(int i = 0 ; i < s ; i++){**

**if(u==arr[i]){**

**System.out.print("Enter new number: ");**

**int nn=in.nextInt();**

**arr[i]=nn;**

**f=1;**

**}**

**}**

**if(f==0){**

**System.out.println("The number you are trying to update doesnt exist");**

**}**

**System.out.println();**

**//Updated Displaying**

**for(int j =0; j <n; j++){**

**System.out.print("| "+j+" |");**

**}**

**System.out.println();**

**for(int i = 0; i<n; i++){**

**System.out.print("| "+arr[i]+" |");**

**}**

**System.out.println("\n");**

**//Sorting**

**System.out.print("The number in Ascending order is");**

**System.out.println("\n");**

**int temp=0;**

**for(int i=0; i<n; i++){**

**for(int j=i+1; j<n; j++){**

**if(arr[i]>arr[j]){**

**temp = arr[i];**

**arr[i] = arr[j];**

**arr[j] = temp;**

**}**

**}**

**}**

**for(int k =0; k <n; k++){**

**System.out.print("| "+k+" |");**

**}**

**System.out.println();**

**for(int i =0; i<n; i++){**

**System.out.print("| "+arr[i]+" |");**

**}**

**System.out.println();**

**//Deleting the array**

**int top=-1;**

**System.out.print("using which style (Q/S): ");**

**n=0;**

**System.out.println(arr[n]);**

**}**

**static void positionInsert(){**

**int pos,num,j,p=3,n;**

**if(p>n){**

**System.out.print("error");**

**System.exit(0);**

**}**

**else{**

**for(int i = 0 ; i < p ; i++){**

**if(i==p){**

**pos=i;**

**j=pos;**

**}**

**}**

**}**

**}**

**static void positionDelete(){**

**int p;**

**}**

**}**

**QUEUE**

**import java.util.\*;**

**public class Queue{**

**int front,rear,max,count;**

**int []a; //array initialization**

**public Queue(int size){**

**max=size;**

**front=0;**

**rear=-1;**

**count=0;**

**a=new int[max]; //a as the array with size max**

**}//queue constructor**

**public void enque(int num){**

**if(rear==max-1){**

**System.out.println("Queue is full");**

**}**

**else{**

**rear++;**

**a[rear]=num;**

**System.out.println("Enque is sucessfull");**

**}//else**

**}//enque**

**public void deque(){**

**if(front>rear){**

**System.out.println("Queue is empty");**

**}**

**else{**

**System.out.println(a[front]+" is deleted");**

**front++;**

**if (front>rear){**

**front=0;**

**rear=-1;**

**}//if**

**}//else**

**}//deque**

**public void InsertFromFront(int numm){**

**if(front==0&&rear==-1){**

**a[front]=numm;**

**rear++;**

**}**

**else if(front!=0){**

**front--;**

**a[front]=numm;**

**}**

**else{**

**System.out.print("insertion cannot be done.");**

**}**

**}**

**public void DeleteFromRear(){**

**if(front>rear){**

**System.out.print("Queue is empty");**

**}**

**else{**

**System.out.print("a["+rear+"]="+a[rear]);**

**rear--;**

**}**

**if(front>rear){**

**front=0;**

**rear=-1;**

**}**

**}**

**public void CircularEnque(int numb){**

**if(count==max){**

**System.out.println("Queue is full");**

**}**

**else{**

**rear=(rear+1)%max;**

**a[rear]=numb;**

**count++;**

**System.out.println("Enque is sucessfull");**

**}//else**

**}//enque**

**public void CircularDeque(){**

**if(count ==0){**

**System.out.println("Queue empty");**

**}**

**else{**

**System.out.println("a[front]is going to be deleted");**

**front=(front+1)%max;**

**count--;**

**}**

**}**

**public void display(){**

**if(front>rear){**

**System.out.println("Queue is empty");**

**}**

**else{**

**System.out.println("values in queue are->");**

**for(int i=front;i<=rear;i++){**

**System.out.println(a[i]+"\t");**

**}**

**}**

**}//display**

**public void DisplayCircle(){**

**int k,j=front;**

**for(k=1;k<=count;k++){**

**System.out.println("values in queue are->");**

**System.out.println(a[j]+"\t");**

**j=(j+1)%max;**

**}**

**}//display circle**

**public static void main(String []arg){**

**Scanner in=new Scanner(System.in);**

**System.out.println("enter size of queue->");**

**int size=in.nextInt();**

**Queue q=new Queue(size);**

**char ch;**

**do{**

**System.out.println("|\_\_\_\_\_\_\_menu\_\_\_\_\_\_\_|");**

**System.out.println("|0|exit |");**

**System.out.println("|1|enque from rear |");**

**System.out.println("|2|deque from front|");**

**System.out.println("|3|display |");**

**System.out.println("|4|enque from front|");**

**System.out.println("|5|deque from rear |");**

**System.out.println("|6|circular enque |");**

**System.out.println("|7|circular deque |");**

**System.out.println("|8|circular display|");**

**System.out.println("|------------------|");**

**System.out.println("\nEnter your choice;");**

**int choice=in.nextInt();**

**switch(choice){**

**case 0:**

**System.exit(0);**

**break;**

**case 1:**

**System.out.println("\nEnter number;");**

**int num=in.nextInt();**

**q.enque(num);**

**break;**

**case 2:**

**q.deque();**

**break;**

**case 3:**

**q.display();**

**break;**

**case 4:**

**System.out.println("\nEnter number;");**

**int numm=in.nextInt();**

**q.InsertFromFront(numm);**

**break;**

**case 5:**

**q.DeleteFromRear();**

**break;**

**case 6:**

**System.out.println("\nEnter number;");**

**int numb=in.nextInt();**

**q.CircularEnque(numb);**

**//q.DisplayCircle();**

**break;**

**case 7:**

**q.CircularDeque();**

**break;**

**case 8:**

**q.DisplayCircle();**

**break;**

**default:**

**System.out.println("wrong choice");**

**}//switch**

**ch='o';**

**}while(ch!='Y');**

**}//main method**

**}//class**

**//finished**

**LINKEDLIST**

**//linked list collection of node which is connected wwith address**

**//lesson6**

**import java.util.\*;**

**class SinglyLinkedList{**

**node start;**

**public SinglyLinkedList(){**

**start=null;**

**}**

**void insertFromFirst(int value){**

**node nod=new node(value);**

**if(start==null){**

**start=nod;**

**start.next=null;**

**//obj.variable=null;**

**//System.out.print();**

**}**

**else{**

**nod.next=start;**

**start=nod;**

**}//els**

**} //void insert from**

**void insertFromBack(int value){**

**node nod=new node(value);**

**if(start==null){**

**start=nod;**

**start.next=null;**

**//obj.variable=null;**

**//System.out.print();**

**}**

**else{**

**node temp=start;**

**while(temp.next!=null){**

**temp=temp.next;**

**}//while**

**temp.next=nod;**

**nod.next=null;**

**}//else**

**} //void insert from**

**void insertFromSpecificPosition(int value,int pos){**

**node nod=new node(value);**

**if(start==null){ //insertiing containig the**

**start=nod;**

**start.next=null;**

**//obj.variable=null;**

**//System.out.print();**

**} //end if**

**else{**

**if(pos==1){**

**nod.next=start;**

**start=nod;**

**} //if**

**else{**

**node temp=null;**

**node temp1=start;**

**int flag=0;**

**for(int i =0 ; i<pos-1;i++){ //check**

**if(temp1!=null){**

**temp=temp1;**

**temp1=temp1.next;**

**}**

**else{**

**System.out.print("position beyond limit");**

**flag=1;**

**break;**

**}**

**}//for**

**if(flag==0){**

**temp.next=nod;**

**nod.next=temp1;**

**}**

**} //else**

**}**

**} //void insert from insertFromSpecificPosition**

**void deleteFromFirst(){**

**if (start==null){**

**System.out.print("LL is empty");**

**}**

**else{**

**node temp=start;**

**start=start.next;**

**System.out.print(temp.data+" is going to be deleted");**

**temp=null;**

**}**

**}**

**void deleteFromLast(){**

**if (start==null){**

**System.out.print("LL is empty");**

**}**

**else{**

**node temp1 = start;**

**node temp = null;**

**while(temp1.next!=null){**

**temp=temp1;**

**temp1=temp.next;**

**}//while**

**System.out.println(temp1.data+ " is going to be deleted");**

**temp1=null;**

**temp.next=null;**

**}**

**}**

**void deleteFromSpecificPosition(int pos){**

**node nod = new node();**

**if(start==null){ //insertiing containig the**

**System.out.print("ll empty");**

**} //end if**

**else{**

**if(pos==1){**

**node temp=start;**

**start=start.next;**

**} //if**

**else{**

**node temp=null;**

**node temp1=start;**

**int flag=0;**

**for(int i =1 ; i<pos;i++){ //check**

**if(temp1!=null){**

**temp=temp1;**

**temp1=temp1.next;**

**}**

**else{**

**System.out.print("position beyond limit");**

**flag=1;**

**break;**

**}**

**}//for**

**if(flag==0){**

**temp.next=temp1.next;**

**System.out.print(temp.data+"is going to be deleted");**

**temp1=null;**

**}**

**} //else**

**}**

**} //void delete from insertFromSpecificPosition**

**void display(){**

**if(start==null){**

**System.out.println("linklist is empty");**

**}**

**else{**

**node temp=start;**

**while(temp!=null){**

**System.out.println(temp.data);**

**temp=temp.next;**

**}//end of while**

**}//end of else statement**

**}//end of display function**

**public static void main(String []args){**

**Scanner in=new Scanner(System.in);**

**SinglyLinkedList SLL =new SinglyLinkedList();**

**int value,pos;**

**char ch;**

**do{**

**System.out.println("\*\*\*\*\*\*\*menu\*\*\*\*\*\*\*");**

**System.out.println("1. insertFromFirst");**

**System.out.println("2. insertFromBack");**

**System.out.println("3. insertFromSpecificPosition");**

**System.out.println("4. display");**

**System.out.println("5. exit");**

**System.out.println("6. delete from first");**

**System.out.println("7. delete from last");**

**System.out.println("8. delete from positon");**

**System.out.println("ENTER THE CHOICE FROM 1 TO 8");**

**int choice = in.nextInt();**

**switch(choice){**

**case 1:**

**System.out.println("Enter a value");**

**value=in.nextInt();**

**SLL.insertFromFirst(value);**

**break;**

**case 2:**

**System.out.println("Enter a value");**

**value=in.nextInt();**

**SLL.insertFromBack(value);**

**break;**

**case 3:**

**System.out.println("Enter a value");**

**value=in.nextInt();**

**System.out.println("Enter a position");**

**pos = in.nextInt();**

**SLL.insertFromSpecificPosition( value, pos);**

**break;**

**case 4:**

**SLL.display();**

**break;**

**case 5:**

**System.exit(0);**

**break;**

**case 6:**

**SLL.deleteFromFirst();**

**break;**

**case 7:**

**SLL.deleteFromLast();**

**break;**

**case 8:**

**System.out.println("Enter a position");**

**pos = in.nextInt();**

**SLL.deleteFromSpecificPosition(pos);**

**break;**

**default:**

**System.out.println(" INVALID CHOICE");**

**}**

**ch ='x';**

**}while(ch!='Y');**

**}**

**}**

**SINGLY CIRCULAR LINKED LIST**

**import java.util.\*;**

**class circularcll{**

**node start;**

**node last;**

**circularcll(){**

**start=null;**

**last=null;**

**}**

**void insertfirst(int value){**

**node nod =new node(value);**

**if(start==null){**

**start=nod;**

**last=nod;**

**last.next=start;**

**}**

**else{**

**nod.next=start;**

**start=nod;**

**last.next=start;**

**}**

**}//void**

**void insertlast(int value){**

**node nod =new node(value);**

**if(start==null){**

**start=nod;**

**last=nod;**

**last.next=start;**

**}**

**else{**

**last.next=nod;**

**last=nod;**

**last.next=start;**

**}**

**}//void**

**void insertspecificposition(int value,int pos){**

**node nod=new node(value);**

**if(start==null){ //insertiing containig the**

**start=nod;**

**last=nod;**

**last.next=start;**

**//obj.variable=null;**

**//System.out.print();**

**} //end if**

**else{**

**if(pos==1){**

**nod.next=start;**

**start=nod;**

**last.next=start;**

**//System.out.print(temp.data+"deleted");**

**//temp=null;**

**} //if**

**else{**

**node temp=null;**

**node temp1=start;**

**int flag=0;**

**for(int i =0 ; i<pos-1;i++){ //check**

**if(temp1!=null){**

**temp=temp1;**

**temp1=temp1.next;**

**}**

**else{**

**System.out.print("position beyond limit");**

**flag=1;**

**break;**

**}**

**}//for**

**if(flag==0){**

**temp.next=nod;**

**nod.next=temp1;**

**}**

**} //else**

**}**

**} //void insert from insertspecificposition**

**void deletefirst(){**

**if (start==null){**

**System.out.print("LL is empty");**

**}**

**else{**

**node temp=start;**

**start=start.next;**

**last.next=start;**

**System.out.print(temp.data+" deleting");**

**temp=null;**

**}**

**}**

**void deletelast(){**

**if (start==null){**

**System.out.print("LL is empty");**

**} //same as ll**

**else if (start == last) {**

**System.out.println(start.data + "is deleted");**

**start= last = null;**

**}**

**else{**

**node temp = start;**

**while(temp.next!=last){**

**temp=temp.next;**

**}//while**

**System.out.print(last.data+ " deleted");**

**last=temp;**

**last.next=start;**

**}**

**}**

**void deletespecificposition(int pos){**

**if(start==null){ //insertiing containig the**

**System.out.print("ll empty");**

**} //end if**

**else{**

**if(pos==1){**

**node temp=start;**

**start=start.next;**

**last.next=start;**

**System.out.print(temp.data+"deleteing");**

**temp=null;**

**last.next=start;**

**} //if**

**else{**

**node temp=null;**

**node temp1=start;**

**int flag=0;**

**for(int i =1 ; i<pos;i++){ //check**

**if(temp1.next!=start){**

**temp=temp1;**

**temp1=temp1.next;**

**}**

**else{**

**System.out.print("position beyond limit");**

**flag=1;**

**break;**

**}**

**}//for**

**if(flag==0){**

**temp.next=temp1.next;**

**System.out.print(temp1.data+"is going to be deleted");**

**temp1=null;**

**}**

**} //else**

**}**

**} //void delete from insertFromSpecificPosition**

**void display() {**

**node temp=start;**

**if(start==null){**

**System.out.println("cll empty");**

**}**

**else{**

**do {**

**System.out.println(temp.data+"\t");**

**temp =temp.next;**

**}while(temp!=start);**

**}**

**}**

**public static void main(String []args){**

**Scanner in=new Scanner(System.in);**

**circularcll cll =new circularcll();**

**int value,pos;**

**char ch;**

**do{**

**System.out.println("\*\*\*\*\*\*\*menu\*\*\*\*\*\*\*");**

**System.out.println("1. insertFromFirst");**

**System.out.println("2. insertFromBack");**

**System.out.println("3. insertFromSpecificPosition");**

**System.out.println("4. display");**

**System.out.println("5. exit");**

**System.out.println("6. delete from first");**

**System.out.println("7. delete from last");**

**System.out.println("8. delete from positon");**

**System.out.println("ENTER THE CHOICE FROM 1 TO 8");**

**int choice = in.nextInt();**

**switch(choice){**

**case 1:**

**System.out.print("Enter a value");**

**value=in.nextInt();**

**cll.insertfirst(value);**

**break;**

**case 2:**

**System.out.print("Enter a value");**

**value=in.nextInt();**

**cll.insertlast(value);**

**break;**

**case 3:**

**System.out.print("Enter a value");**

**value=in.nextInt();**

**System.out.print("Enter a position");**

**pos = in.nextInt();**

**cll.insertspecificposition( value, pos);**

**break;**

**case 4:**

**cll.display();**

**break;**

**case 5:**

**System.exit(0);**

**break;**

**case 6:**

**cll.deletefirst();**

**break;**

**case 7:**

**cll.deletelast();**

**break;**

**case 8:**

**System.out.println("Enter a position");**

**pos = in.nextInt();**

**cll.deletespecificposition(pos);**

**break;**

**default:**

**System.out.println(" INVALID CHOICE");**

**}**

**ch ='x';**

**}while(ch!='Y');**

**}**

**}**

**DOUBLY LINKED LIST**

**import java.util.\*;**

**class DoublyLinkedList{**

**node start=null;**

**void insertFromFirst(int value){**

**node nod=new node(value);**

**if(start==null){ //khali**

**start=nod; // node lai start**

**start.next=null; //start ko next le null**

**start.prev=null; //start ko previous le null**

**}**

**else{**

**nod.next=start; //node ko next le start**

**start.prev=nod;**

**start=nod;**

**start.prev=null;**

**}**

**}//void insert form first**

**void insertFromLast(int values){**

**node nod=new node(values);**

**if(start==null){**

**start=nod;**

**start.next=null;**

**start.prev=null;**

**}**

**else{**

**node temp=start;**

**while(temp.next!=null){**

**temp=temp.next;**

**}**

**temp.next=nod;**

**nod.next=null;**

**nod.prev=temp;**

**}**

**}//void insert form last**

**void deleteFromFirst(){**

**if(start==null){**

**System.out.println("Linklist is empty");**

**}**

**else if (start.next == null) {**

**System.out.println(start.data + " is deleted");**

**start = null;**

**}**

**else{**

**node temp=start;**

**start=start.next;**

**start.prev=null;**

**System.out.print(temp.data+"deleting");**

**temp=null;**

**}**

**}//void delete form first**

**void deleteFromLast(){**

**if(start==null){**

**System.out.println("Linklist is empty");**

**}**

**else if (start.next == null) {**

**System.out.println(start.data + " is deleted");**

**start= null;**

**}**

**else{**

**node temp1=start;**

**node temp=null;**

**while(temp1.next!=null){**

**temp=temp1;**

**temp1=temp.next;**

**}**

**temp.next=null;**

**System.out.print(temp1.data+"deleting");**

**temp1 = null;**

**}**

**}//void delete form last**

**void insertspecificposition(int value,int pos){**

**node nod =new node(value);**

**if(start==null){ //insertiing containig the**

**start=nod;**

**start.next=null;**

**start.prev=null;**

**} //end if**

**else{**

**if(pos==1){**

**nod.next=start;**

**start.prev=nod;**

**start=nod;**

**nod.prev=null;**

**//System.out.print(temp.data+"deleted");**

**//temp=null;**

**} //if**

**else{**

**node temp=null;**

**node temp1=start;**

**int flag=0;**

**for(int i =0 ; i<pos-1;i++){ //check**

**if(temp1!=null){**

**temp=temp1;**

**temp1=temp.next;**

**}**

**else{**

**System.out.print("position beyond limit");**

**flag=1;**

**break;**

**}**

**}//for**

**if(flag==0){**

**if(temp.next==null){**

**temp.next=nod;**

**nod.prev=temp;**

**}**

**else{**

**temp.next = nod;**

**nod.next = temp1;**

**nod.prev = temp;**

**temp1.prev = nod;**

**}**

**}**

**} //else**

**}**

**} //void insert from insertspecificposition**

**void deletespecificposition(int pos){**

**node nod=new node();**

**if(start==null){ //insertiing containig the**

**System.out.println("LinkedList is Empty");**

**} //end if**

**else{**

**if(pos==1){**

**if(start.next==null){**

**System.out.print(start.data +"deleted");**

**}else{**

**node temp=start;**

**start=start.next;**

**start.prev=null;**

**System.out.print(temp.data+"deleting");**

**temp = null;**

**} //if**

**}**

**else{**

**node temp=null;**

**node temp1=start;**

**int flag=0;**

**for(int i =0 ; i<pos-1;i++){ //check**

**if(temp1.next!=null){**

**temp=temp1;**

**temp1=temp.next;**

**}**

**else{**

**System.out.print("position beyond limit");**

**flag=1;**

**break;**

**}**

**}//for**

**if(flag==0){**

**temp.next=temp1.next;**

**temp1.next.prev=temp;**

**System.out.print(temp1.data+"deleting");**

**temp1=null;**

**}**

**} //else**

**}**

**} //void insert from deletespecificposition**

**void display(char n) {**

**if(n=='f'){**

**if (start == null) {**

**System.out.println("LinkedList is empty");**

**} else {**

**node temp = start ;**

**while (temp != null) {**

**System.out.print(temp.data + "\t");**

**temp = temp.next;**

**}**

**}**

**}**

**else{**

**if (start == null) {**

**System.out.println("LinkedList is empty");**

**} else {**

**node temp = start ;**

**while(temp.next!=null){**

**temp=temp.next;**

**}**

**while (temp != null) {**

**System.out.print(temp.data + "\t");**

**temp = temp.prev;**

**}**

**}**

**}**

**}**

**public static void main(String []args){**

**Scanner in=new Scanner(System.in);**

**DoublyLinkedList cll =new DoublyLinkedList();**

**int value,pos;**

**char ch;**

**do{**

**System.out.println("\*\*\*\*\*\*\*menu\*\*\*\*\*\*\*");**

**System.out.println("1. insertFromFirst");**

**System.out.println("2. insertFromBack");**

**System.out.println("3. insertFromSpecificPosition");**

**System.out.println("4. display");**

**System.out.println("5. exit");**

**System.out.println("6. delete from first");**

**System.out.println("7. delete from last");**

**System.out.println("8. delete from positon");**

**System.out.println("ENTER THE CHOICE FROM 1 TO 8");**

**int choice = in.nextInt();**

**switch(choice){**

**case 1:**

**System.out.println("Enter a value");**

**value=in.nextInt();**

**cll.insertFromFirst(value);**

**break;**

**case 2:**

**System.out.println("Enter a value");**

**value=in.nextInt();**

**cll.insertFromLast(value);**

**break;**

**case 3:**

**System.out.println("Enter a value");**

**value=in.nextInt();**

**System.out.println("Enter a position");**

**pos = in.nextInt();**

**cll.insertspecificposition( value, pos);**

**break;**

**case 4:**

**System.out.println("display front/back");**

**char n =in.next().charAt(0);**

**cll.display(n);**

**break;**

**case 5:**

**System.exit(0);**

**break;**

**case 6:**

**cll.deleteFromLast();**

**break;**

**case 7:**

**cll.deleteFromFirst();**

**break;**

**case 8:**

**System.out.println("Enter a position");**

**pos = in.nextInt();**

**cll.deletespecificposition(pos);**

**break;**

**default:**

**System.out.println(" INVALID CHOICE");**

**}**

**ch ='x';**

**}while(ch!='Y');**

**}**

**}**

**Circulardoubly**

**import java.util.\*;**

**public class DCLL{**

**node start;**

**node last;**

**public DCLL(){**

**start=null;**

**last=null;**

**}**

**public void insertFirst(int value){**

**node nod =new node(value);**

**if(start==null){**

**start=last=nod;**

**last.next=start;**

**start.prev=last;**

**}**

**else{**

**nod.next=start;**

**start.prev=nod;**

**start=nod;**

**start.prev=last;**

**last.next=start;**

**}**

**}**

**public void insertLast(int value){**

**node nod =new node(value);**

**if(start==null){**

**start=last=nod;**

**last.next=start;**

**start.prev=last;**

**}**

**else{**

**nod.prev=last;**

**last.next=nod;**

**last=nod;**

**last.next=start;**

**start.prev=last;**

**}**

**}**

**public void insertSpecific(int value, int pos){**

**node nod =new node(value);**

**if(start==null){**

**start=last=nod;**

**last.next=start;**

**start.prev=last;**

**}**

**else if(pos==1){**

**nod.next=start;**

**start.prev=nod;**

**start=nod;**

**start.prev=last;**

**last.next=start;**

**}**

**else{**

**node temp= start;**

**node temp1= start.next;**

**int flag=0;**

**for(int i=1;i<pos-1;i++){**

**if(temp1!=start){**

**temp=temp1;**

**temp1=temp.next;**

**}**

**else{**

**flag=1;**

**System.out.print("position beyond limit");**

**break;**

**}**

**}**

**if(flag==0){**

**temp.next=nod;**

**nod.prev=temp;**

**nod.next=temp1;**

**temp1.prev=nod;**

**}**

**}**

**}**

**public void deleteFirst(){**

**//node nod =new node();**

**if( start == null){**

**System.out.print("empty");**

**}**

**else if( start == last){**

**System.out.println("deleting");**

**start=last=null;**

**}**

**else{**

**node temp=start;**

**start=start.next;**

**start.prev = last;**

**last.next=start;**

**System.out.println(temp.data+"deleting");**

**temp=null;**

**}**

**}**

**public void deleteLast(){**

**//node nod =new node();**

**if(start==null){**

**System.out.print("empty");**

**}**

**else if(start==last){**

**System.out.println("deleting");**

**start=last=null;**

**}**

**else{**

**/\***

**node temp=last.prev;**

**start.prev=temp;**

**last=temp;**

**\*/**

**node temp=last.prev;**

**temp.next=start;**

**start.prev=temp;**

**System.out.println(last.data+"deleting");**

**last=null;**

**last=temp;**

**}**

**}//method**

**public void deleteSpecific(int pos){**

**//node nod =new node();**

**if(start==null){**

**System.out.print("empty");**

**}//if**

**else if(start==last){**

**System.out.print(start.data+"deleted");**

**start=last=null;**

**}**

**else if(pos==1){**

**node temp=start;**

**start=start.next;**

**start.prev=last;**

**last.next=start;**

**System.out.println(temp.data+"deleting");**

**temp=null;**

**} //else if**

**else{**

**node temp= start;**

**node temp1= start.next;**

**int flag=0;**

**for(int i=1;i<pos-1;i++){**

**if(temp1!= start){**

**temp=temp1;**

**temp1=temp.next;**

**} //if**

**else{**

**flag=1;**

**System.out.print("position beyond limit");**

**break;**

**} //else**

**} //for**

**if(flag==0){**

**temp.next = temp1.next;**

**temp1.next.prev = temp;**

**System.out.println(temp1.data+" is deleting");**

**temp1 = null;**

**} //if**

**} //else**

**} //method**

**void display(int b) {**

**if(b==1&&start!=null){**

**node temp = start;**

**do{**

**System.out.print(temp.data + "\t");**

**temp = temp.next;**

**}while(temp != start);**

**System.out.println();**

**}**

**else if(b==2&&start!=null) {**

**node temp1 = last;**

**do{**

**System.out.print(temp1.data+"\t");**

**temp1 = temp1.prev;**

**}while(temp1 != last);**

**System.out.println();**

**}**

**if(start==null){**

**System.out.println("LinkedList is empty");**

**}**

**}**

**public static void main(String[] args) throws Exception{**

**Scanner in = new Scanner(System.in);**

**DCLL dl = new DCLL();**

**int value, pos;**

**char y;**

**do {**

**System.out.println("Choice MENU ");**

**System.out.println(" 1 -> INSERT FROM FIRST");**

**System.out.println(" 2 -> INSERT FROM LAST");**

**System.out.println(" 3 -> INSERT FROM SPECIFIC POSITION");**

**System.out.println(" 4 -> DELETE FROM FIRST");**

**System.out.println(" 5 -> DELETE FROM LAST");**

**System.out.println(" 6 -> DELETE FROM SPECIFIC POSITION");**

**System.out.println(" 7 -> DISPLAY");**

**System.out.println(" 8 -> EXIT");**

**System.out.print("Choice [1 - 8]: ");**

**int choice = in.nextInt();**

**switch (choice) {**

**case 1:**

**System.out.print("Insert Value: ");**

**value = in.nextInt();**

**dl.insertFirst(value);**

**break;**

**case 2:**

**System.out.print("Insert Value: ");**

**value = in.nextInt();**

**dl.insertLast(value);**

**break;**

**case 3:**

**System.out.print("Enter value: ");**

**value = in.nextInt();**

**System.out.print("Enter Position: ");**

**pos = in.nextInt();**

**dl.insertSpecific(value, pos);**

**break;**

**case 4:**

**dl.deleteFirst();**

**break;**

**case 5:**

**dl.deleteLast();**

**break;**

**case 6:**

**System.out.print("Enter Position: ");**

**pos = in.nextInt();**

**dl.deleteSpecific(pos);**

**break;**

**case 7:**

**System.out.println("type 1-front/2-back:");**

**int b=in.nextInt();**

**System.out.println("Elements are:");**

**dl.display(b);**

**break;**

**case 8:**

**System.exit(100);**

**Thread.sleep(1000);**

**break;**

**default:**

**System.out.println("Invalid Choice !!!");**

**}**

**y = '\*';**

**} while (y != 'x');**

**} //main**

**}//class**

**factorial**

**import java.util.\*;**

**public class Factorial {**

**static int fact(int n){**

**if(n==0)**

**return 1;**

**else{**

**return (n\*fact(n-1));**

**}**

**}**

**public static void main(String[] args) {**

**Scanner scan=new Scanner(System.in);**

**int fact=1;**

**System.out.println ("Enter the number to calculate factorial");**

**int num=scan.nextInt(); // It is number to calculate factorial**

**fact=fact(num);**

**System.out.println ("Factorial of "+num+" is:"+fact);**

**}**

**}**

**add**

**import java.util.\*;**

**class add{**

**public static void main(String[] args) {**

**Scanner in = new Scanner(System.in);**

**int n = 10;**

**// int sum = 0;**

**System.out.print("enter number=");**

**n=in.nextInt();**

**sum(n,0,1);**

**}**

**public static void sum(int n,int summ,int i){**

**summ=summ+i;**

**i++;**

**sum(n,summ,i);**

**if( i<=n ){**

**System.exit(0);**

**}**

**}**

**}**

**powers**

**import java.util.Scanner;**

**import java.util.\*;**

**public class Powers {**

**public static void main(String[] args) {**

**int a;**

**int n;**

**int res;**

**Scanner in = new Scanner(System.in);**

**System.out.print("Enter int a ");**

**a = in.nextInt();**

**System.out.print("Enter int n ");**

**n = in.nextInt();**

**res = Powers.pow(a, n);**

**System.out.print(res);**

**}**

**public static int pow(int a, int n) {**

**int result = 1;**

**if(n==1){**

**result = a;**

**} else{**

**result = a \* pow(a, n - 1);**

**}**

**return result;**

**}**

**}**

**tower of Hanoi**

**import java.util.\*;**

**//lab finished done**

**class towerOfHanoi{ //in order disPlay of tree**

**public static void main(String args[]){**

**int n,count=0;**

**Scanner scan=new Scanner(System.in);**

**do{**

**System.out.print("enter the no of disk::");**

**n=scan.nextInt();**

**hanoi(n,'A','B','C');**

**count=2\* (int)(Math.pow(2,n-1))-1;**

**System.out.println("number to do it="+count+"\n");**

**}while(n!=-1);**

**}**

**//static int count=0;**

**public static void hanoi( int n, char src, char spare, char dest){**

**if(n==0)**

**System.exit(0);**

**if( n==1 ){**

**System.out.println("transfering disc 1 from peg "+src+" to peg "+dest);**

**}**

**else{**

**hanoi(n-1,src,dest,spare);**

**System.out.println("transfering disc "+n+" from peg "+src+" to peg "+dest);**

**hanoi( n-1,spare,src,dest);**

**}**

**}**

**}**

**//2^n-1 ac ab ac**

**/\***

**search**

**linear first to last index search one by one**

**binary value takes check range 1-50 num>/<25**

**a-b a-c a-b a-c**

**pre-in-post order tree**

**NLR-LNR-LRN**

**3,2,1,1,2,1,1-1,2,1,3,1,2,1-1,1,2,1,1,2,3**

**\*/**

**fibonanci**

**import java.util.\*;**

**class Fibonacci{**

**public static void main(String args[])**

**{**

**Scanner in = new Scanner(System.in);**

**int n,i,count;**

**System.out.print("enter number=");**

**count=in.nextInt();**

**//System.out.print(n1+" "+n2);//printing 0 and 1**

**for(i=0;i<count;++i)//loop starts from 2 because 0 and 1 are already printed**

**{**

**System.out.print(fib(i) +"\0");**

**}**

**}**

**public static int fib(int i){**

**if(i==0)**

**return 0;**

**else if(i==1)**

**return 1;**

**else**

**return fib(i-1)+fib(i-2);**

**}**

**}**

**bubble sort**

**package Sorting;**

**import java.util.Scanner;**

**public class BubbleSort {**

**public static void main(String[] args) {**

**Scanner scan = new Scanner(System.in);**

**int size;**

**int[] num = new int[20];**

**System.out.println("Enter the size of an array::");**

**size = scan.nextInt();**

**for (int i = 0; i < size; i++) {**

**System.out.print("\nEnter " + (i + 1) + " number::");**

**num[i] = scan.nextInt();**

**}**

**int temp;**

**for (int i = 0; i < size; i++) {**

**for (int j = 0; j < size - i - 1; j++) {**

**if (num[j] > num[j + 1]) {**

**temp = num[j];**

**num[j] = num[j + 1];**

**num[j + 1] = temp;**

**}**

**}**

**}//end of for loop**

**System.out.println("Sorted Element are::");**

**for (int i = 0; i < size; i++) {**

**System.out.print(num[i]+"; ");**

**}**

**}//end of main function**

**}//end of class**

**insertion sort**

**//package Sorting;**

**import java.util.Scanner;**

**public class InsertionSort {**

**public static void main(String[] args) {**

**Scanner scan = new Scanner(System.in);**

**int size;**

**int[] num = new int[20];**

**System.out.println("Enter the size of an array::");**

**size = scan.nextInt();**

**int temp, j;**

**for (int i = 0; i < size; i++) {**

**System.out.print("\nEnter " + (i + 1) + " number::");**

**num[i] = scan.nextInt();**

**// }**

**// int temp, j;**

**// for (int i = 1; i < size; i++) {**

**temp = num[i];**

**j = i - 1;**

**try {**

**while ((temp < num[j]) && (j >= 0)) {**

**num[j + 1] = num[j];**

**j = j - 1;**

**}**

**} catch (ArrayIndexOutOfBoundsException e) {**

**// System.out.println("Caught" + e.getMessage());**

**}**

**num[j + 1] = temp;**

**}//end of for loop**

**System.out.println("Sorted Element are::");**

**for (int i = 0; i < size; i++) {**

**System.out.print(num[i] + "; ");**

**}**

**}//end of main function**

**}**

**selection sort**

**package Sorting;**

**import java.util.Scanner;**

**public class SelectionSort {**

**void selection(int num[], int size) {**

**int loc, temp;**

**int min = num[0];**

**for (int i = 0; i < size; i++) {**

**min = num[i];**

**loc = i;**

**for (int j = i + 1; j < size; j++) {**

**if (num[j] < min) {**

**min = num[j];**

**loc = j;**

**}**

**}**

**if (loc != i) {**

**temp = num[i];**

**num[i] = num[loc];**

**num[loc] = temp;**

**}**

**}//end of loop**

**}//end of function**

**public static void main(String[] args) {**

**Scanner scan = new Scanner(System.in);**

**SelectionSort sort=new SelectionSort();**

**int size;**

**int[] num = new int[20];**

**System.out.println("Enter the size of an array::");**

**size = scan.nextInt();**

**for (int i = 0; i < size; i++) {**

**System.out.print("\nEnter " + (i + 1) + " number::");**

**num[i] = scan.nextInt();**

**}**

**sort.selection(num, size);**

**System.out.println("Sorted Element are::");**

**for (int i = 0; i < size; i++) {**

**System.out.print(num[i]+"; ");**

**}**

**}**

**}**