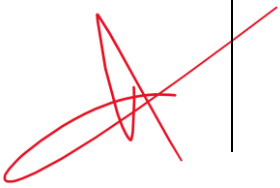
Create a class Array which contains an integer array to hold n number of data. Perform the following operation on Array instance:

i) Replace First
ii) ReplaceAnyPos
iii) Search
iv) Sort
v) Traversal

Source Code:

```
import java.util.Scanner;
public class Array {
    int n;
    int[] a;
    // Constructor to initialize the array
    public Array(int n) {
        this.n = n;
        a = new int[n];
    }
    // Replace the first element
    void ReplaceFirst(int e) {
        a[0] = e;
    }
    // Replace an element at a specific position
    void ReplaceAnyPos(int p, int v) {
        if (p >= 0 && p < n) {
            a[p] = v;
        } else {
            System.out.println("Invalid position.");
        }
    }
    // Search for an element in the array
    void Search(int e) {
        boolean found = false;
```

```
        for (int i = 0; i < n; i++) {
            if (a[i] == e) {
                found = true;
                break;
            }
        }
        if (found) {
            System.out.println("Element Available");
        } else {
            System.out.println("Element Unavailable");
        }
    }
    // Sort the array in ascending order
    void Sort() {
        for (int i = 0; i < n; i++) {
            for (int j = i + 1; j < n; j++) {
                if (a[i] > a[j]) {
                    int temp = a[i];
                    a[i] = a[j];
                    a[j] = temp;
                }
            }
        }
    }
    // Traverse and print the array elements
    void Traversal() {
        System.out.println("Array elements:");
        for (int i = 0; i < n; i++) {
            System.out.print(a[i] + " ");
        }
        System.out.println();
    }
```

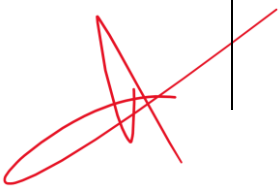



```
}
class ArrayDemo {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the size of the array: ");
        int size = sc.nextInt();
        Array arr = new Array(size);
        System.out.println("Enter " + size + " elements:");
        for (int i = 0; i < size; i++) {
            arr.a[i] = sc.nextInt();
        }
        int choice;
        do {
            System.out.println("\nMenu:");
            System.out.println("1. Replace First");
            System.out.println("2. Replace Any Position");
            System.out.println("3. Search");
            System.out.println("4. Sort");
            System.out.println("5. Traverse");
            System.out.println("6. Exit");
            System.out.print("Enter your choice: ");
            choice = sc.nextInt();
            switch (choice) {
                case 1:
                    System.out.print("Enter the element to replace the first element: ");
                    int firstElement = sc.nextInt();
                    arr.ReplaceFirst(firstElement);
                    System.out.println("First element replaced.");
                    break;
                case 2:
                    System.out.print("Enter the position to replace: ");
                    int position = sc.nextInt();
```

```
                System.out.print("Enter the value to insert: ");
                int value = sc.nextInt();
                arr.ReplaceAnyPos(position, value);
                break;
            case 3:
                System.out.print("Enter the element to search: ");
                int searchElement = sc.nextInt();
                arr.Search(searchElement);
                break;
            case 4:
                arr.Sort();
                System.out.println("Array sorted.");
                break;
            case 5:
                arr.Traversal();
                break;
            case 6:
                System.out.println("Program Closed");
                break;
            default:
                System.out.println("Invalid choice!");
        }
    } while (choice != 6);
    sc.close();
  }
}
```


Output:

```
Enter the size of the array: 5
Enter 5 elements:
95 96 97 98 99

Menu:
1. Replace First
2. Replace Any Position
3. Search
4. Sort
5. Traverse
6. Exit
Enter your choice: 1
Enter the element to replace the first element: 1
First element replaced.

Menu:
1. Replace First
2. Replace Any Position
3. Search
4. Sort
5. Traverse
6. Exit
Enter your choice: 2
Enter the position to replace: 2
Enter the value to insert: 3

Menu:
1. Replace First
2. Replace Any Position
3. Search
```

4. Sort

5. Traverse

6. Exit

Enter your choice: 3

Enter the element to search: 2

Element Unavailable

Menu:

1. Replace First

2. Replace Any Position

3. Search

4. Sort

5. Traverse

6. Exit

Enter your choice: 4

Array sorted.

Menu:

1. Replace First

2. Replace Any Position

3. Search

4. Sort

5. Traverse

6. Exit

Enter your choice: 5

Array elements:

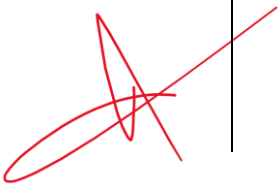
1 3 96 98 99

Menu:

1. Replace First

2. Replace Any Position

3. Search

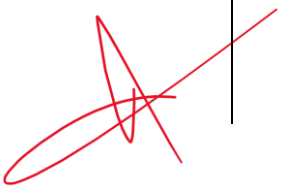


```
4. Sort
5. Traverse
6. Exit
Enter your choice: 6
Program Closed
```


Implementation of Stack by LinkedList.

Source Code:

```
import java.util.Scanner;
public class LLStack {
   private java.util.LinkedList list=new java.util.LinkedList();
   public LLStack() {
   }
   public void clear(){
      list.clear();
   }
   public boolean isEmpty() {
      return list.isEmpty();
   }
   public Object topE1() {
      if(isEmpty())
      {
         throw new java.util.EmptyStackException();
      }
      return list.getLast();
   }
   public Object pop()
   {
      if(isEmpty())
      {
         throw new java.util.EmptyStackException();
      }
      return list.removeLast();
   }
   public void push(Object el)
   {
      list.addLast(el);
```

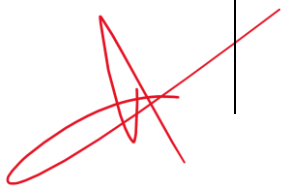



```
    }
    public String toString()
    {
        return list.toString();
    }
}
class LLStackDemo {
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        LLStack list=new LLStack();
        while(true)
        {
            System.out.println("Enter choice: \n\n1.Push\t2.Pop\t3.Print top element\t4.Exit");
            int choice=sc.nextInt();
            switch(choice)
            {
                case 1:
                    System.out.println("Enter element to push in the stack:");
                    int el=sc.nextInt();
                    list.push(el);
                    break;
                case 2:
                    System.out.println(list.pop()+"deleted from the stack.");
                    break;
                case 3:
                    System.out.println("Top element:"+list.topE1());
                    break;
                case 4:
                    System.out.println("Exit Successfully");
                    return;
                default:
                    System.out.println("Invalid Choice");
```

```
        }
      }
    }
}
```

Output:

```
Enter choice:


1.Push    2.Pop     3.Print top element          4.Exit

1

Enter element to push in the stack:

3

Enter choice:


1.Push    2.Pop     3.Print top element          4.Exit

1

Enter element to push in the stack:

4

Enter choice:


1.Push    2.Pop     3.Print top element          4.Exit

1

Enter element to push in the stack:

5

Enter choice:


1.Push    2.Pop     3.Print top element          4.Exit

3

Top element:5

Enter choice:


1.Push    2.Pop     3.Print top element          4.Exit
```

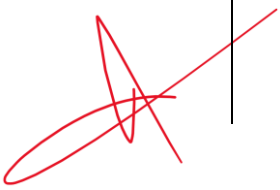


```
2
5deleted from the stack.
Enter choice:


1.Push    2.Pop     3.Print top element          4.Exit
4
Exit Successfully
```


Implementation of Stack by Array.

Source Code:

```
import java.util.Scanner;
public class StackArray {
    Integer arr[];
    int tos;
    public StackArray(int size) {
        tos=-1;
        arr=new Integer[size];
    }
    public boolean isFull() {
        return(tos==arr.length-1);
    }
    public boolean isEmpty() {
        return (tos==-1);
    }
    public void push(Integer el) {
        if(isFull())
            System.out.println("Stack OverFlow Error");
        else
        {
            tos++;
            arr[tos]=el;
        }
    }
    public Integer pop() {
        if(isEmpty())
        {
            System.out.println("Stack UnderFlow Error");
            return null;
        }
```

```
        else
            return arr[tos--];
    }
    public Integer peek() {
        return arr[tos];
    }
}
class StackDemo {
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        System.out.print("Enter the size of stack:");
        int size=sc.nextInt();
        StackArray stack=new StackArray(size);
        while(true)
        {
            System.out.print("1.Push\t2.Pop\t3.Print top element\t4.Exit\nEnter your choice:");
            int choice=sc.nextInt();
            switch(choice)
            {
                case 1:
                    System.out.print("Enter element to push in the stack:");
                    int el=sc.nextInt();
                    stack.push(el);
                    break;
                case 2:
                    System.out.println(stack.pop()+" deleted from the stack.");
                    break;
                case 3:
                    System.out.println("Top element:"+stack.peek());
                    break;
                case 4:
                    System.out.println("Exit Successfully");
```

```
                return;
            default:
                System.out.println("Invalid Choice");
        }
    }
  }
}
```

Output:

```
Enter the size of stack:3
1.Push  2.Pop  3.Print top element  4.Exit
Enter your choice:1
Enter element to push in the stack:10
1.Push  2.Pop  3.Print top element  4.Exit
Enter your choice:1
Enter element to push in the stack:20
1.Push  2.Pop  3.Print top element  4.Exit
Enter your choice:3
Top element:20
1.Push  2.Pop  3.Print top element  4.Exit
Enter your choice:2
20 deleted from the stack.
1.Push  2.Pop  3.Print top element  4.Exit
Enter your choice:3
Top element:10
1.Push  2.Pop  3.Print top element  4.Exit
Enter your choice:4
Exit Successfully
```


Write a program to demonstrate Array List.

Source Code:

```
import java.util.ArrayList;
import java.util.Scanner;
public class ArrayListDemo {
    public static void main(String[] args) {
        ArrayList<String> std=new ArrayList<String>();
        std.add("Ashesh");
        std.add("Bishow");
        std.add("Sanyam");
        std.add("Parbat");
        System.out.println(std);
        std.add(0,"Amit");
        System.out.println(std.get(0));
        std.set(0,"Nikesh");
        std.remove(0);
        std.clear();
        System.out.println(std.size());
        System.out.println(std);
    }
}
```

Output:

```
[Ashesh, Bishow, Sanyam, Parbat]
Amit
0
[]
```

Write a program to demonstrate Linked List.

Source Code:

```
import java.util.LinkedList;
import java.util.Scanner;
public class LinkedListDemo {
    public static void main(String[] args) {
        LinkedList<String> LL = new LinkedList<String>();
        LL.add("I");
        LL.add("am");
        LL.add("Ashesh");
        System.out.println(LL);
        System.out.println(LL.size());
        LL.add(0,"Neupane");
        System.out.println(LL.get(0));
        LL.set(0,"HighApproach");
        LL.remove(0);
        LL.clear();
        System.out.println(LL);
    }
}
```

Output:

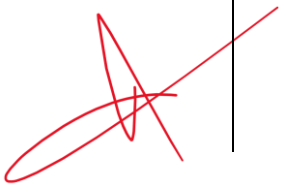
```
[I, am, Ashesh]
3
Neupane
[]
```


Write a program to demonstrate Doubley Linked List.

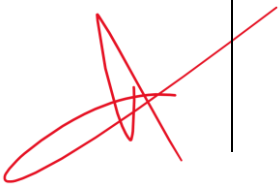
Source Code:

```
import java.util.Scanner;
class Node {
    Integer data;
    Node prev,next;
    public Node(Node prev,int data,Node next) {
        this.data=data;
        this.next=next;
        this.prev=prev;
    }
}
public class DLL {
    Node head,tail;
    public boolean isEmpty() {
        return (head==null && tail==null);
    }
    public void delTail() {
        if(isEmpty())
            System.out.println("You cannot delete from empty list");
        else if(head==tail)
            head=tail=null;
        else
        {
            tail=tail.prev;
            tail.next=null;
        }
    }
    public void delHead() {
        if(isEmpty())
            System.out.println("You cannot delete from empty list");
```

```
    else if(head==tail)
        head=tail=null;
    else
    {
        head=head.next;
        head.prev=null;
    }
}
public void insertHead(int el) {
    if(isEmpty())
        head=tail=new Node(null,el,null);
    else
    {
        Node nn=new Node(null,el,head);
        nn.next.prev=nn;
        head=nn;
    }
}
public void insertTail(int el) {
    if(isEmpty())
        head=tail=new Node(null,el,null);
    else
    {
        Node nn=new Node(tail,el,null);
        tail.next=nn;
        tail=nn;
    }
}
public void printForward() {
    Node temp=head;
    while(temp!=null)
    {
```

```
            System.out.print(temp.data+"->");
            temp=temp.next;
        }
        System.out.println("null");
    }
    public void printReverse() {
        Node temp=tail;
        while(temp!=null)
        {
            System.out.print(temp.data+"->");
            temp=temp.prev;
        }
        System.out.println("null");
    }
    public boolean Search(int key) {
        boolean found=false;
        Node temp=head;
        if(!isEmpty())
        {
            while(temp!=null)
            {
                if(key==temp.data)
                {
                    found=true;
                    break;
                }
                temp=temp.next;
            }
        }
        return found;
    }
}
```

```
class DLLDemo {
   public static void main(String[] args) {
      DLL dll=new DLL();
      Scanner sc=new Scanner(System.in);
      while(true)
      {
         System.out.print("1.Add First\t2.Add Last\t3.Print Forward\t4.Print Reverse\t5.Search\t6.Exit\nEnter a choice:");
         int choice=sc.nextInt();
         switch(choice)
         {
            case 1:
               System.out.print("Enter a number to insert:");
               int el=sc.nextInt();
               dll.insertHead(el);
               break;
            case 2:
               System.out.print("Enter a number to insert:");
               el=sc.nextInt();
               dll.insertTail(el);
               break;
            case 3:
               dll.printForward();
               break;
            case 4:
               dll.printReverse();
               break;
            case 5:
               System.out.print("Enter a number to search:");
               int key=sc.nextInt();
               System.out.println(dll.Search(key));
               break;
```

```
            case 6:
                System.out.println("Exit Successful");
                return;
            default:
                System.out.println("Invalid Choice");
                break;
        }
    }
  }
}
```

Output:

```
1.Add First     2.Add Last     3.Print Forward   4.Print Reverse   5.Search 6.Exit
Enter a choice:1
Enter a number to insert:3
1.Add First     2.Add Last     3.Print Forward   4.Print Reverse   5.Search 6.Exit
Enter a choice:1
Enter a number to insert:4
1.Add First     2.Add Last     3.Print Forward   4.Print Reverse   5.Search 6.Exit
Enter a choice:1
Enter a number to insert:5
1.Add First     2.Add Last     3.Print Forward   4.Print Reverse   5.Search 6.Exit
Enter a choice:2
Enter a number to insert:9
1.Add First     2.Add Last     3.Print Forward   4.Print Reverse   5.Search 6.Exit
Enter a choice:3
5->4->3->9->null
1.Add First     2.Add Last     3.Print Forward   4.Print Reverse   5.Search 6.Exit
Enter a choice:5
Enter a number to search:3
true
1.Add First     2.Add Last     3.Print Forward   4.Print Reverse   5.Search 6.Exit
```

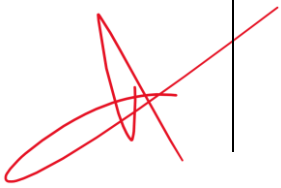
```
Enter a choice:4
9->3->4->5->null
1.Add First     2.Add Last     3.Print Forward   4.Print Reverse   5.Search 6.Exit
Enter a choice:6
Exit Successful
```


Write a program to demonstrate SinglyLinked List.

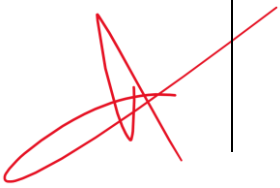
Source Code:

```
import java.util.Scanner;
class Node {
    int data;
    Node next;
    public Node(int data,Node next) {
        this.data=data;
        this.next=next;
    }
}
public class SLL {
    Node head,tail;
    public SLL() {
        head=tail=null;
    }
    public Boolean isEmpty() {
        return(head==null && tail==null);
    }
    public void insertAtLast(int el) {
        if(isEmpty())
            head=tail=new Node(el,null);
    }
    public void insertAtFirst(int el) {
        if(isEmpty())
            head=tail=new Node(el,null);
        else
            head=new Node(el,head);
    }
    public void DeleteFirst(){
        if(isEmpty())
```

```
        System.out.println("LinkedList is empty");
    else if(head==tail)
        head=tail=null;
    else
        head=head.next;
}
public void DeleteLast() {
    if(isEmpty())
        System.out.println("LinkedListis empty");
    else if(head==tail)
        head=tail=null;
    else
    {
        Node temp=head;
        while(temp.next!=tail)
        {
            temp=temp.next;
        }
        temp.next=null;
        tail=temp;
    }
}
public boolean Search(int key) {
    boolean found=false;
    Node temp=head;
    if(!isEmpty())
    {
        while(temp!=null)
        {
            if(key==temp.data)
            {
                found=true;
```

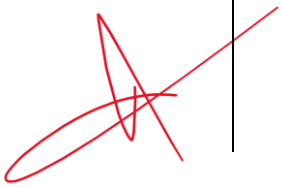
```
                break;
            }
            temp=temp.next;
        }
    }
    return found;
}
public void traverse() {
    Node temp=head;
    if(!isEmpty())
    {
        while(temp!=null)
        {
            System.out.print(temp.data+"->");
            temp=temp.next;
        }
    }
}
public void PutAnyPos(int el,int pos) {
    if(isEmpty())
    {
        if(pos==0)
            head=tail=new Node(el,null);
        else
            System.out.println("Invalid Position");
    }
    else
    {
        Node temp=head;
        if(pos==0)
            insertAtFirst(el);
        else
```

```
        {
            for(int i=0;i<pos-1;i++)
            {
                if(temp==null)
                    break;
                temp=temp.next;
            }
            if(temp==null)
                System.out.println("Invalid Position");
            else if(temp==tail)
                insertAtLast(el);
            else
                temp.next=new Node(el,temp.next);
        }
    }
}
public void DeleteAnyPos(int pos) {
    if(isEmpty())
        System.out.println("List is empty");
    else if(pos==0)
    {
        head=head.next;
        if(head==null)
        tail=null;
    }
    else
    {
        Node temp=head;
        boolean validPos=true;
        for(int i=0;i<pos-1;i++)
        {
            if(temp==null||temp.next==null)
```

```
            {
                System.out.println("Invalid Position");
                validPos=false;
                break;
            }
            temp=temp.next;
        }
        if(validPos)
        {
            if(temp==null||temp.next==null)
                System.out.println("Invalid Position");
            else
            {
                temp.next=temp.next.next;
                if(temp.next==null)
                    tail=temp;
            }
        }
    }
  }
}
class SLLDemo {
   public static void main(String[] args) {
      SLL sl=new SLL();
      Scanner sc=new Scanner(System.in);
      while(true)
      {
         System.out.print("1.Add First\t2.Add Last\t3.Insert At Any Position\t4.Delete First\t5.Delete Last\t6. Delete Any Position\t7.Search\t8.Traverse\t9.Exit\nEnter a choice:");
         int choice=sc.nextInt();
         switch(choice)
         {
```

```
case 1:
    System.out.print("Enter a number to insert:");
    int el=sc.nextInt();
    sl.insertAtFirst(el);
    break;
case 2:
    System.out.print("Enter a number to insert:");
    el=sc.nextInt();
    sl.insertAtLast(el);
    break;
case 3:
    System.out.print("Enter a number to insert & Position:");
    el=sc.nextInt();
    int pos=sc.nextInt();
    sl.PutAnyPos(el, pos);
    break;
case 4:
    sl.DeleteFirst();
    break;
case 5:
    sl.DeleteLast();
    break;
case 6:
    System.out.print("Enter the position:");
    pos=sc.nextInt();
    sl.DeleteAnyPos(pos);
    break;
case 7:
    System.out.print("Enter a number to search:");
    int key=sc.nextInt();
    System.out.println(sl.Search(key));
    break;
```



```
            case 8:
                sl.traverse();
                System.out.println();
                break;
            case 9:
                System.out.println("Exit Successful");
                return;
            default:
                System.out.println("Invalid Choice");
        }
    }
  }
}
```

Output:

```
1.Add First    2.Add Last    3.Insert At Any Position    4.Delete First    5.Delete Last    6. Delete Any Position  7.Search  8.Traverse    9.Exit

Enter a choice:1

Enter a number to insert:1

1.Add First    2.Add Last    3.Insert At Any Position    4.Delete First    5.Delete Last    6. Delete Any Position  7.Search  8.Traverse    9.Exit

Enter a choice:1

Enter a number to insert:2

1.Add First    2.Add Last    3.Insert At Any Position    4.Delete First    5.Delete Last    6. Delete Any Position  7.Search  8.Traverse    9.Exit

Enter a choice:1

Enter a number to insert:3

1.Add First    2.Add Last    3.Insert At Any Position    4.Delete First    5.Delete Last    6. Delete Any Position  7.Search  8.Traverse    9.Exit

Enter a choice:2

Enter a number to insert:5

1.Add First    2.Add Last    3.Insert At Any Position    4.Delete First    5.Delete Last    6. Delete Any Position  7.Search  8.Traverse    9.Exit

Enter a choice:3
```

```
Enter a number to insert & Position:2 4

Invalid Position

1.Add First 2.Add Last 3.Insert At Any Position 4.Delete First 5.Delete Last 6. Delete Any Position 7.Search 8.Traverse 9.Exit

Enter a choice:3

Enter a number to insert & Position:4 2

1.Add First 2.Add Last 3.Insert At Any Position 4.Delete First 5.Delete Last 6. Delete Any Position 7.Search 8.Traverse 9.Exit

Enter a choice:4

1.Add First 2.Add Last 3.Insert At Any Position 4.Delete First 5.Delete Last 6. Delete Any Position 7.Search 8.Traverse 9.Exit

Enter a choice:5

1.Add First 2.Add Last 3.Insert At Any Position 4.Delete First 5.Delete Last 6. Delete Any Position 7.Search 8.Traverse 9.Exit

Enter a choice:7

Enter a number to search:2

true

1.Add First 2.Add Last 3.Insert At Any Position 4.Delete First 5.Delete Last 6. Delete Any Position 7.Search 8.Traverse 9.Exit

Enter a choice:8

2->4->

1.Add First 2.Add Last 3.Insert At Any Position 4.Delete First 5.Delete Last 6. Delete Any Position 7.Search 8.Traverse 9.Exit

Enter a choice:9

Exit Successful
```

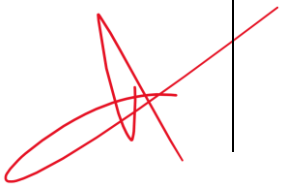

Write a program to demonstrate Circular Linked List.

Source Code:

```
import java.util.Scanner;


class CLNode {
    int data;
    CLNode next;

    public CLNode(int data) {
        this.data = data;
        this.next = null;
    }

    public CLNode(int data, CLNode next) {
        this.data = data;
        this.next = next;
    }
}

public class CLList {
    CLNode head, tail;

    public CLList() {
        head = tail = null;
    }

    public boolean isEmpty() {
        return (head == null && tail == null);
    }

    public void insertAtHead(int el) {
```

```
    if (isEmpty()) {
        head = tail = new CLNode(el);
        tail.next = head;
    } else {
        CLNode nn = new CLNode(el, head);
        tail.next = nn;
        head = nn;
    }
}

public void insertAtTail(int el) {
    if (isEmpty()) {
        head = tail = new CLNode(el);
        tail.next = head;
    } else {
        CLNode nn = new CLNode(el, head);
        tail.next = nn;
        tail = nn;
    }
}

public void insertAtAnyPos(int el, int pos) {
    if (pos <= 0) {
        System.out.println("Position must be positive");
        return;
    }
    if (pos == 1) {
        insertAtHead(el);
        return;
    }
    CLNode temp = head;
    int count = 1;
```

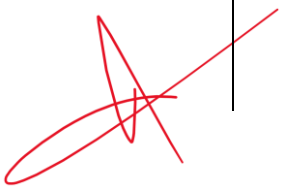
```
    while (temp != tail && count < pos - 1) {
        temp = temp.next;
        count++;
    }
    if (count == pos - 1) {
        CLNode nn = new CLNode(el, temp.next);
        temp.next = nn;
        if (temp == tail)
            tail = nn;
    } else {
        System.out.println("Invalid Position");
    }
}

public void deleteHead() {
    if (isEmpty())
        System.out.println("LinkedList is Empty");
    else if (head == tail)
        head = tail = null;
    else {
        head = head.next;
        tail.next = head;
    }
}

public void deleteTail() {
    if (isEmpty())
        System.out.println("LinkedList is empty");
    else if (head == tail)
        head = tail = null;
    else {
        CLNode temp = head;
```

```
        while (temp.next != tail) {
            temp = temp.next;
        }
        temp.next = head;
        tail = temp;
    }
}

public void deleteAtAnyPos(int pos) {
    if (isEmpty() || pos <= 0) {
        System.out.println("Invalid Position or list is empty");
        return;
    }
    if (pos == 1) {
        deleteHead();
        return;
    }
    CLNode temp = head;
    int count = 1;
    while (temp.next != head && count < pos - 1) {
        temp = temp.next;
        count++;
    }
    if (count == pos - 1 && temp.next != null) {
        if (temp.next == tail)
            tail = temp;
        temp.next = temp.next.next;
    } else {
        System.out.println("Invalid Position");
    }
}
```

```
    public boolean Search(int el) {
        if (isEmpty())
            return false;
        CLNode temp = head;
        do {
            if (temp.data == el)
                return true;
            temp = temp.next;
        } while (temp != head);
        return false;
    }

    public void Traverse() {
        if (isEmpty()) {
            System.out.println("List is empty");
            return;
        }
        CLNode temp = head;
        do {
            System.out.print(temp.data + "->");
            temp = temp.next;
        } while (temp != head);
        System.out.println("Back to Head");
    }
}

class CircularLDemo {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        CLList list = new CLList();
```

```
System.out.println("1.InsertAtHead\t2.InsertAtTail\t3.InsertAtAnyPos\t4.DeleteHead\t5.DeleteTail\t6.DeleteAnyPos
\t7.Search\t8.Traverse\t9.Exit\nProgram will only close if entered 9");
        while (true) {
            System.out.print("Enter your choice: ");
            int choice = sc.nextInt();
            switch (choice) {
                case 1:
                    System.out.print("Enter element: ");
                    int el = sc.nextInt();
                    list.insertAtHead(el);
                    break;
                case 2:
                    System.out.print("Enter element: ");
                    el = sc.nextInt();
                    list.insertAtTail(el);
                    break;
                case 3:
                    System.out.print("Enter element and position: ");
                    el = sc.nextInt();
                    int pos = sc.nextInt();
                    list.insertAtAnyPos(el, pos);
                    break;
                case 4:
                    list.deleteHead();
                    break;
                case 5:
                    list.deleteTail();
                    break;
                case 6:
                    System.out.print("Enter position: ");
                    pos = sc.nextInt();
```

```
                list.deleteAtAnyPos(pos);
                break;
            case 7:
                System.out.print("Enter element to search: ");
                el = sc.nextInt();
                System.out.println("Found: " + list.Search(el));
                break;
            case 8:
                list.Traverse();
                break;
            case 9:
                System.out.println("Exit Successfully");
                return;
            default:
                System.out.println("Invalid Choice");
        }
    }
  }
}
```

Output:

```
1.InsertAtHead  2.InsertAtTail  3.InsertAtAnyPos4.DeleteHead  5.DeleteTail  6.DeleteAnyPos  7.Search
8.Traverse  9.Exit
Program will only close if entered 9
Enter your choice: 1
Enter element: 5
Enter your choice: 1
Enter element: 10
Enter your choice: 1
Enter element: 15
Enter your choice: 2
Enter element: 20
Enter your choice: 8
```

```
15->10->5->20->Back to Head
Enter your choice: 7
Enter element to search: 5
Found: true
Enter your choice: 3
Enter element and position: 2 8
Invalid Position
Enter your choice: 3
Enter element and position: 8 2
Enter your choice: 6
Enter position: 3
Enter your choice: 8
15->8->5->20->Back to Head
Enter your choice: 9
Exit Successfully
```


Write a program to add two large integers using stack.

Source Code:

```
import java.util.Stack;
public class LargeNumericOperation {
   public static String addLargeNumbers(String num1,String num2){
      Stack<Integer> stack1=new Stack<>();
      Stack<Integer> stack2=new Stack<>();
      for(int i=0;i<num1.length();i++){
         stack1.push(num1.charAt(i)-'0');
      }
      for(int i=0;i<num2.length();i++){
         stack2.push(num2.charAt(i)-'0');
      }
      Stack<Integer> resultstack=new Stack<>();
      int carry=0;
      while(!stack1.isEmpty()||!stack2.isEmpty()||carry!=0){
         int sum=carry;
         if(!stack1.isEmpty()){
            sum+=stack1.pop();
         }
         if(!stack2.isEmpty()){
            sum+=stack2.pop();
         }
         carry=sum/10;
         resultstack.push(sum%10);
      }
      StringBuilder result=new StringBuilder();
      while(!resultstack.isEmpty()){
         result.append(resultstack.pop());
      }
      return result.toString();
```

```
    }
    public static void main(String[] args){
        String num1="9876543210";
        String num2="1123456789";
        String result=addLargeNumbers(num1,num2);
        System.out.println("Sum:"+result);
    }
}
```

Output:

```
Sum:10999999999
```


Write a program to convert and print infix expression to postfix expression.

Source Code:

```
import java.util.Stack;
import java.util.Scanner;
public class PostfixDemo {
    public static int precedence(char ch) {
        if (ch == '+' || ch == '-') return 1;
        else if (ch == '*' || ch == '/') return 2;
        return -1;
    }
    public static boolean isOperator(char ch) {
        return ch == '+' || ch == '-' || ch == '*' || ch == '/';
    }
    public static void main(String[] args) {
        Stack<Character> opstack = new Stack<>();
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter an arithmetical infix expression:");
        String infix = sc.nextLine().replaceAll("\\s+", "");
        String postfix = "";
        char arr[] = infix.toCharArray();
        for (int i = 0; i < arr.length; i++) {
            char ch = arr[i];
            if (Character.isLetterOrDigit(ch)) postfix += ch;
            else if (ch == '(') opstack.push(ch);
            else if (isOperator(ch)) {
                while (!opstack.isEmpty() && isOperator(opstack.peek()) && precedence(opstack.peek()) >=
precedence(ch)) {
                    postfix += opstack.pop();
                }
                opstack.push(ch);
            } else if (ch == ')') {
                while (!opstack.isEmpty() && opstack.peek() != '(') {
```

```
                postfix += opstack.pop();
            }
            if (!opstack.isEmpty() && opstack.peek() == '(') opstack.pop();
        }
    }
    while (!opstack.isEmpty()) postfix += opstack.pop();
    System.out.println("The expression in postfix is:" + postfix);
  }
}
```

Output:

```
Enter an arithmetical infix expression:(A+B)*C-(D+E)*(F+G)
The expression in postfix is:AB+C*DE+FG+*-
```


Write a program to evaluate the postfix expression.

Source Code:

```
import java.util.Scanner;
import java.util.Stack;
public class PostFix {
    public static int evaluatePostfix(String expression){
        Stack<Integer> stack=new Stack<>();
        for(int i=0;i<expression.length();i++){
            char ch=expression.charAt(i);
            if(Character.isDigit(ch)){
                stack.push(ch-'0');
            }
            else if(ch=='+'||ch=='-'||ch=='*'||ch=='/'){
                int operand2=stack.pop();
                int operand1=stack.pop();
                int result=0;
                switch(ch){
                    case '+':
                        result=operand1+operand2;
                        break;
                    case '-':
                        result=operand1-operand2;
                        break;
                    case '*':
                        result=operand1*operand2;
                        break;
                    case '/':
                        result=operand1/operand2;
                        break;
                }
                stack.push(result);
```

```
            }
        }
        return stack.pop();
    }
    public static void main(String[] args){
        Scanner sc=new Scanner(System.in);
        String postfixexp;
        System.out.println("Enter the Postfix Expression:");
        postfixexp=sc.next();
        int result=evaluatePostfix(postfixexp);
        System.out.println("Result:"+result);
    }
}
```

Output:

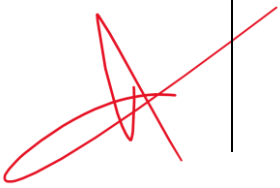
```
Enter the Postfix Expression:
12+3*45-67+*-
Result:22
```


Write a program to convert and print infix expression to prefix expression.

Source Code:

```
import java.util.Scanner;
import java.util.Stack;
public class InfixToPrefix {
    public static int precedence(char ch){
        if(ch=='+'||ch=='-')
            return 1;
        else
            return 2;
    }
    public static boolean isOperator(char ch){
        if(ch=='+'||ch=='-'||ch=='*'||ch=='/')
            return true;
        return false;
    }
    public static void main(String[] args){
        Scanner sc=new Scanner(System.in);
        Stack<Character> opstack=new Stack<>();
        String infix;
        System.out.println("Enter an arithmetical infix expression:");
        infix=sc.next();
        String prefix="";
        String postfix="";
        char arr[]=infix.toCharArray();
        for(int i=0;i<arr.length;i++)
        {
            char ch=arr[i];
            if(ch=='(')
                opstack.push(ch);
            else if(isOperator(ch))
```

```
        {
            if(opstack.empty())
            {
                opstack.push(ch);
                continue;
            }
            while(!opstack.isEmpty()){
                if(isOperator(opstack.peek())){
                    if(precedence(opstack.peek())>precedence(ch)){
                        postfix+=opstack.pop();
                    }
                    else
                        break;
                }
                else
                    break;
            }
            opstack.push(ch);
        }
        else if(ch==')'){
            while(opstack.peek()!='('){
                postfix+=opstack.pop();
            }
            opstack.pop();
        }
        else
            postfix+=ch;
    }
    while(!opstack.empty()){
        postfix+=opstack.pop();
    }
    for(int i=postfix.length()-1;i>=0;i--){
```

```
            prefix+=postfix.charAt(i);
        }
        System.out.println("The expression in prefix is:"+prefix);
    }
}
```

Output:

```
Enter an arithmetical infix expression:
(A+B)*C-(D-E)*(F+G)
The expression in prefix is:-*+GF-ED*C+BA
```


Write a program to evaluate the prefix expression.

Source Code:

```
import java.util.Scanner;
import java.util.Stack;
public class PrefixDemo {
    public static int evaluatePrefix(String expression){
        Stack<Integer> stack=new Stack<>();
        for(int i=expression.length()-1;i>=0;i--){
            char ch=expression.charAt(i);
            if(Character.isDigit(ch)){
                stack.push(ch-'0');
            }
            else if(ch=='+'||ch=='-'||ch=='*'||ch=='/'){
                int operand2=stack.pop();
                int operand1=stack.pop();
                int result=0;
                switch(ch){
                    case '+':
                        result=operand1+operand2;
                        break;
                    case '-':
                        result=operand1-operand2;
                        break;
                    case '*':
                        result=operand1*operand2;
                        break;
                    case '/':
                        result=operand1/operand2;
                        break;
                }
                stack.push(result);
```

```
            }
        }
        return stack.pop();
    }
    public static void main(String[] args){
        Scanner sc=new Scanner(System.in);
        String prefixexp;
        System.out.println("Enter the Prefix Expression:");
        prefixexp=sc.next();
        int result=evaluatePrefix(prefixexp);
        System.out.println("Result:"+result);
    }
}
```

Output:

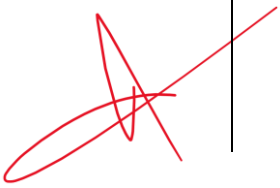
```
Enter the Prefix Expression:
-*+76-54*3+21
Result:22
```


Write a program to show all the operations in queue using array.

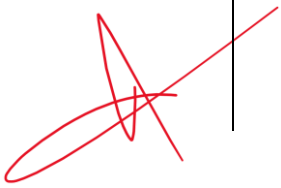
Source Code:

```
import java.util.Scanner;
public class QueueUsingArray {
    int front,rear;
    Integer arr[];
    int size=0;
    public QueueUsingArray(int s){
        size=s;
        arr=new Integer[s];
        front=rear=-1;
    }
    public boolean isFull(){
        return(rear==size-1);
    }
    public boolean isEmpty(){
        return(front==-1&&rear==-1);
    }
    public void enqueue(int el){
        if(isFull())
            System.out.println("Queue is full");
        else
        {
            if(isEmpty())
            {
                front++;
                rear++;
                arr[rear]=el;
            }
            else
            {
```

```
            rear++;
            arr[rear]=el;
        }
    }
}
public Integer dequeue(){
    Integer element=null;
    if(isEmpty())
    {
        System.out.println("Cannot Delete from the queue");
    }
    else if(front==rear)
    {
        element=arr[front];
        front=rear=-1;
        return element;
    }
    else
    {
        element=arr[front];
        front++;
        System.out.println("Deleted Successfully");
    }
    return element;
}
public Integer getFirst() throws Exception{
    return arr[front];
}
public Integer getLast() throws Exception{
    return arr[rear];
}
public void printAll(){
```

```
        int temp=front;
        System.out.println("The elements in the stack are:");
        if(!isEmpty())
        {
            while(temp!=rear)
            {
                System.out.println(arr[temp]);
                temp++;
            }
            System.out.println(arr[temp]);
        }
        else
            System.out.println("There is no element to search");
    }
}
class QueueDemo{
    public static void main(String[] args){
        Scanner sc=new Scanner(System.in);
        QueueUsingArray q=new QueueUsingArray(5);
        System.out.println("1.Enqueue an element\n2.Dequeue an element\n3.Get the first element\n4.Get the last element\n5.Print all the elements\n6.Exit");
        while(true){
            System.out.print("Enter a choice:");
            int choice=sc.nextInt();
            switch(choice)
            {
                case 1:
                    System.out.print("Enter an element:");
                    int el=sc.nextInt();
                    q.enqueue(el);
                    break;
                case 2:
```

```
                Integer el1=q.dequeue();
                if(el1!=null)
                    System.out.println("The deleted element is:"+el1);
                break;
            case 3:
                try{
                    System.out.println("The first element is:"+q.getFirst());
                }
                catch(Exception e){
                    System.out.println("There is no element in the stack");
                }
                break;
            case 4:
                try{
                    System.out.println("The last element is:"+q.getLast());
                }
                catch(Exception e){
                    System.out.println("There is no element in the stack");
                }
                break;
            case 5:
                q.printAll();
                break;
            case 6:
                System.out.println("Exit Successfully");
                return;
            default:
                System.out.println("Invalid Choice");
        }
    }
  }
}
```

Output:

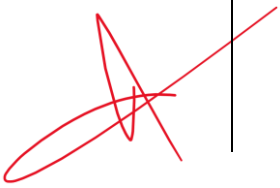
```
1.Enqueue an element
2.Dequeue an element
3.Get the first element
4.Get the last element
5.Print all the elements
6.Exit
Enter a choice:1
Enter an element:1
Enter a choice:1
Enter an element:2
Enter a choice:1
Enter an element:3
Enter a choice:2
Deleted Successfully
The deleted element is:1
Enter a choice:3
The first element is:2
Enter a choice:4
The last element is:3
Enter a choice:5
The elements in the stack are:
2
3
Enter a choice:6
Exit Successfully
```


Write a program to show all the operations in circular queue using array.

Source Code:

```
import java.util.Scanner;
public class CircularQueue {
    int front,rear;
    Integer arr[];
    int size=0;
    public CircularQueue(int s){
        size=s;
        arr=new Integer[s];
        front=rear=-1;
    }
    public boolean isFull(){
        return((front==0&&rear==size-1)||(front==rear+1));
    }
    public boolean isEmpty(){
        return(front==-1&&rear==-1);
    }
    public void enqueue(int el){
        if(isFull())
            System.out.println("Queue is full");
        else{
            if(isEmpty()){
                front++;
                rear++;
                arr[rear]=el;
            }
            else{
                rear=(rear+1)%size;
                arr[rear]=el;
            }
```

```
    }
}
public Integer dequeue(){
    Integer element=null;
    if(isEmpty())
        System.out.println("Cannot delete from an empty queue");
    else if(front==rear){
        element=arr[front];
        front=rear=-1;
        return element;
    }
    else{
        element=arr[front];
        front=(front+1)%size;
        System.out.println("Deleted Successfully");
    }
    return element;
}
public Integer getFirst()throws Exception{
    return arr[front];
}
public Integer getLast()throws Exception{
    return arr[rear];
}
public void printAll(){
    int temp=front;
    if(!isEmpty()){
        while(temp!=rear){
            System.out.println(arr[temp]);
            temp=(temp+1)%size;
        }
        System.out.println(arr[temp]);
```

```
        }
        else
            System.out.println("There is no element to print");
    }
}
class CircularQueueDemo{
    public static void main(String[] args){
        Scanner sc=new Scanner(System.in);
        CircularQueue q=new CircularQueue(5);
        System.out.println("1.Enqueue an element\n2.Dequeue an element\n3.Get the first element\n4.Get the last element\n5.Print all the element\n6.Exit");
        while(true){
            System.out.print("Enter a choice:");
            int choice=sc.nextInt();
            switch(choice)
            {
                case 1:
                    System.out.print("Enter an element:");
                    int el=sc.nextInt();
                    q.enqueue(el);
                    break;
                case 2:
                    Integer el1=q.dequeue();
                    if(el1!=null)
                        System.out.println("The deleted element is:"+el1);
                    break;
                case 3:
                    try{
                        System.out.println("The first element is:"+q.getFirst());
                    }
                    catch(Exception e){
                        System.out.println("There is no element in the stack");
```

```
                }
                break;
            case 4:
                try{
                    System.out.println("The last element is:"+q.getLast());
                }
                catch(Exception e){
                    System.out.println("There is no element in the stack");
                }
                break;
            case 5:
                q.printAll();
                break;
            case 6:
                System.out.println("Exit Successfully");
                return;
            default:
                System.out.println("Invalid Choice");
        }
    }
  }
}
```

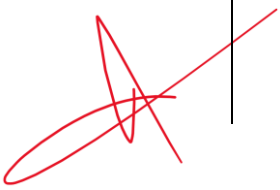

Output:

```
1.Enqueue an element
2.Dequeue an element
3.Get the first element
4.Get the last element
5.Print all the element
6.Exit
Enter a choice:1
Enter an element:1
Enter a choice:1
Enter an element:2
Enter a choice:1
Enter an element:3
Enter a choice:2
Deleted Successfully
The deleted element is:1
Enter a choice:3
The first element is:2
Enter a choice:4
The last element is:3
Enter a choice:5
2
3
Enter a choice:6
Exit Successfully
```

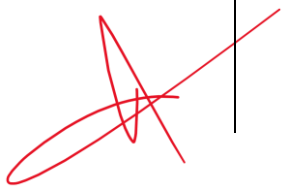

Write a program to show all the operations of queue using Linked List.

Source Code:

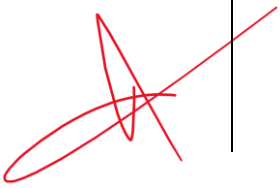
```
import java.util.Scanner;
public class QueueUsingLL {
    Node front,rear;
    public QueueUsingLL(){
        rear=front=null;
    }
    public boolean isEmpty(){
        return(front==null&&rear==null);
    }
    public void enqueue(int el){
        if(isEmpty())
            front=rear=new Node(el,null);
        else
            rear=rear.next=new Node(el,rear);
    }
    public Integer dequeue(){
        Integer element=null;
        if(isEmpty())
            System.out.println("Cannot delete from the queue");
        else if(front==rear){
            element=front.data;
            front=rear=null;
            System.out.println("Deleted Successfully");
            return element;
        }
        else{
            element=front.data;
            front=front.next;
            System.out.println("Deleted Successfully");
```

```
        }
        return element;
    }
    public Integer getFirst() throws Exception{
        return front.data;
    }
    public Integer getLast() throws Exception{
        return rear.data;
    }
    public void printAll(){
        Node temp=front;
        if(!isEmpty()){
            System.out.println("The elements are:");
            while(temp!=rear){
                System.out.println(temp.data);
                temp=temp.next;
            }
            System.out.println(temp.data);
        }
        else
            System.out.println("There is no element to search");
    }
}
class QueueDemo{
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        QueueUsingLL q=new QueueUsingLL();
        System.out.println("1.Enqueue an element\n2.Dequeue an element\n3.Get the first element\n4.Get the last element\n5.Print all elements\n6.Exit");
        while(true){
            System.out.print("Enter a choice:");
            int choice=sc.nextInt();
```



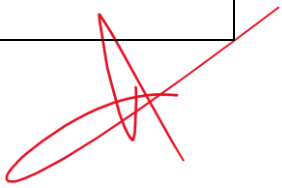
```
switch(choice){
    case 1:
        System.out.print("Enter an element:");
        int el=sc.nextInt();
        q.enqueue(el);
        break;
    case 2:
        Integer el1=q.dequeue();
        if(el1!=null)
            System.out.println("The deleted element is:"+el1);
        break;
    case 3:
        try{
            System.out.println("The first element is:"+q.getFirst());
        }
        catch(Exception e){
            System.out.println("There is no element in the queue");
        }
        break;
    case 4:
        try{
            System.out.println("The last element is:"+q.getLast());
        }
        catch(Exception e){
            System.out.println("There is no element in the queue");
        }
        break;
    case 5:
        q.printAll();
        break;
    case 6:
        System.out.println("Exit Successfully");
```

```
                return;
            default:
                System.out.println("Invalid Choice");
        }
    }
  }
}
```

Output:

```
1.Enqueue an element
2.Dequeue an element
3.Get the first element
4.Get the last element
5.Print all elements
6.Exit
Enter a choice:1
Enter an element:1
Enter a choice:1
Enter an element:2
Enter a choice:2
Deleted Successfully
The deleted element is:1
Enter a choice:3
The first element is:2
Enter a choice:4
The last element is:2
Enter a choice:5
The elements are:
2
Enter a choice:6
Exit Successfully
```

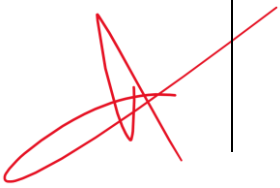



Write a program to show all operations of circular queue using Linked List.

Source Code:

```
import java.util.Scanner;
class Node1 {
    Node1 next;
    int data;
    public Node1(int d,Node1 n){
        data=d;
        next=n;
    }
}
class CircularQueue{
    Node1 front,rear;
    public CircularQueue(){
        front=rear=null;
    }
    public boolean isEmpty(){
        return(front==null&&rear==null);
    }
    public void enqueue(int el){
        if(isEmpty()){
            front=rear=new Node1(el,null);
        }
        else
        {
            rear=rear.next=new Node1(el,rear);
            rear.next=front;
        }
    }
    public Integer dequeue(){
        Integer element=null;
```

```
    if(isEmpty()){
        System.out.println("Cannot delete from the queue");
    }
    else if(front==rear){
        element=front.data;
        front=rear=null;
        System.out.println("Deleted Successfully");
        return element;
    }
    else{
        element=front.data;
        front=front.next;
        System.out.println("Deleted Successfully");
    }
    return element;
}
public Integer getFirst() throws Exception{
    return front.data;
}
public Integer getLast() throws Exception{
    return rear.data;
}
public void printAll(){
    Node1 temp=front;
    if(!isEmpty()){
        System.out.println("The elements are:");
        while(temp.next!=front){
            System.out.println(temp.data);
            temp=temp.next;
        }
    }
    else
```

```
            System.out.println("There is no element to print");
        }
}
public class CircularQueueDemo{
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        CircularQueue q=new CircularQueue();
        System.out.println("1.Enqueue an element\n2.Dequeue an element\n3.Get the first element\n4.Get the last element\n5.Print all the elements\n6.Exit");
        while(true){
            System.out.print("Enter a choice:");
            int choice=sc.nextInt();
            switch(choice){
                case 1:
                    System.out.print("Enter an element:");
                    int el=sc.nextInt();
                    q.enqueue(el);
                    break;
                case 2:
                    Integer el1=q.dequeue();
                    if(el1!=null)
                        System.out.println("The deleted element is:"+el1);
                    break;
                case 3:
                    try{
                        System.out.println("The first element is:"+q.getFirst());
                    }
                    catch(Exception e){
                        System.out.println("There is no element in the queue");
                    }
                    break;
                case 4:
```

```
                try{
                    System.out.println("The last element is:"+q.getLast());
                }
                catch(Exception e){
                    System.out.println("There is no element in the queue");
                }
                break;
            case 5:
                q.printAll();
                break;
            case 6:
                System.out.println("Exit Successfully");
                return;
            default:
                System.out.println("Invalid Choice");
        }
    }
  }
}
```


Output:

```
1.Enqueue an element
2.Dequeue an element
3.Get the first element
4.Get the last element
5.Print all the elements
6.Exit
Enter a choice:1
Enter an element:1
Enter a choice:1
Enter an element:2
Enter a choice:1
Enter an element:3
Enter a choice:2
Deleted Successfully
The deleted element is:1
Enter a choice:3
The first element is:2
Enter a choice:4
The last element is:3
Enter a choice:5
The elements are:
2
3
Enter a choice:6
Exit Successfully
```


Write a program to print numbers from 1 to n using recursion.

Source Code:

```
import java.util.Scanner;
public class Print
{
   public static void main(String[] args)
   {
      Scanner sc=new Scanner(System.in);
      System.out.print("Enter the value of n:");
      int n=sc.nextInt();
      Print ob=new Print();
      ob.PrintNumbers(n,1);
   }
   void PrintNumbers(int n, int i)
   {
      if(i<=n)
      {
         System.out.print(i+" ");
         PrintNumbers(n,i+1);
      }
   }
}
```

Output:

```
Enter the value of n:5
1 2 3 4 5
```


Write a program to print numbers from n to 1 using recursion.

Source Code:

```
import java.util.Scanner;
public class Print
{
   public static void main(String[] args)
   {
      Scanner sc=new Scanner(System.in);
      System.out.print("Enter the value of n:");
      int n=sc.nextInt();
      Print ob=new Print();
      ob.PrintNumbers(n,1);
   }
   void PrintNumbers(int n, int i)
   {
      if(i<=n)
      {
         System.out.print(n+" ");
         PrintNumbers(n-1,i);
      }
   }
}
```

Output:

```
Enter the value of n:5
5 4 3 2 1
```


Write a program to print sum of natural numbers upto n by using recursion.

Source Code:

```
import java.util.Scanner;
public class Print
{
   public static void main(String[] args)
   {
      Scanner sc=new Scanner(System.in);
      System.out.print("Enter the value of n:");
      int n=sc.nextInt();
      Print ob=new Print();
      System.out.print("The sum of natural number is: "+ob.sum(n));
   }
   public int sum(int n)
   {
      if(n==1)
      return 1;
      else
      return n+sum(n-1);
   }
}
```

Output:

```
Enter the value of n:10
The sum of natural number is: 55
```


Write a program to print nth term of fibonacci series.

Source Code:

```
import java.util.Scanner;
public class FiboSeriesnth {
    public int fibo(int n){
        if(n==1)
            return 0;
        else if(n==2)
            return 1;
        else
            return (fibo(n-1)+fibo(n-2));
    }
}
class FiboDemo{
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        System.out.print("Enter the nth term of fibonacci series:");
        int n=sc.nextInt();
        FiboSeriesnth p=new FiboSeriesnth();
        System.out.println("The fibonacci number is:"+p.fibo(n));
    }
}
```

Output:

```
Enter the nth term of fibonacci series:9
The fibonacci number is:21
```


Write a program to print the factorial of a number using recursion.

Source Code:

```
import java.util.Scanner;
public class FactorialRecursion {
    public int fact(int n){
        if(n==0||n==1)
            return 1;
        else
            return n*fact(n-1);
    }
}
class FactDemo{
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        FactorialRecursion f=new FactorialRecursion();
        System.out.print("Enter a number to find it's factorial:");
        int n=sc.nextInt();
        System.out.println("The factorial of the given number is:"+f.fact(n));
    }
}
```

Output:

```
Enter a number to find it's factorial:5
The factorial of the given number is:120
```


Write a program to print the factorial of a number using tail recursion.

Source Code:

```
import java.util.Scanner;
public class FactorialTailRec {
    int i=1;
    public int fact(int n){
        if(n==1)
            return i;
        else
        {
            i=i*n;
            this.i=i;
            return fact(n-1);
        }
    }
}
class FactDemo{
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        FactorialTailRec f=new FactorialTailRec();
        System.out.print("Enter a number to find it's factorial:");
        int n=sc.nextInt();
        System.out.println("The factorial of the given number is:"+f.fact(n));
    }
}
```

Output:

```
Enter a number to find it's factorial:5
The factorial of the given number is:120
```

Write a program to find the product of two numbers using addition operator only.

Source Code:

```
import java.util.Scanner;
class Multiplication
{
   public static void main(String[] args)
   {
      Scanner sc=new Scanner(System.in);
      System.out.print("Enter any two numbers to find it's product:");
      int x=sc.nextInt();
      int y=sc.nextInt();
      Multiplication ob=new Multiplication();
      System.out.print("Result is: "+ob.mul(x,y));
   }
   int mul(int x, int y)
   {
      if(x==1)
      return y;
      else if(y==1)
      return x;
      else
      return x+mul(x,y-1);
   }
}
```

Output:

```
Enter any two numbers to find it's product:2 3
Result is: 6
```

Write a program to check whether the number is prime or not using recursion.

Source Code:

```
import java.util.Scanner;
public class PrimeCheck
{
   public static void main(String[] args)
   {
      Scanner sc=new Scanner(System.in);
      System.out.print("Enter any number: ");
      int n=sc.nextInt();
      PrimeCheck ob=new PrimeCheck();
      if (n<=1)
      {
         System.out.println(n + " is not a prime number");
      }
      else
      {
         int x=ob.prime(n,2);
         if(x==1)
         System.out.println(n + " is a prime number");
         else
         System.out.println(n + " is not a prime number");
      }
   }
   int prime(int y, int i)
   {
      if (i*i>y)
      {
         return 1;
      }
      else if (y%i==0)
```

```
        {
            return 0;
        }
        else
        {
            return prime(y,i+1);
        }
    }
}
```

Output:

```
Enter any number: 17
17 is a prime number
```


Write a program to compute power using product operator.

Source Code:

```
import java.util.Scanner;
public class PowerCompute {
    public int power(int a,int b){
        if(b==1)
            return a;
        else
            return a*power(a,b-1);
    }
}
class RecDemo{
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        System.out.print("Enter base:");
        int a=sc.nextInt();
        System.out.print("Enter power:");
        int b=sc.nextInt();
        PowerCompute p=new PowerCompute();
        int power=p.power(a, b);
        System.out.println("Result:"+power);
    }
}
```

Output:

```
Enter base:4
Enter power:2
Result:16
```


Write a program to demonstrate nested recursion.

Source Code:

```
public class NestedRecursionDemo {
    public static int nestedRecursion(int n) {
        if (n > 100)
            return n - 10;
        return nestedRecursion(nestedRecursion(n + 11));
    }
    public static void main(String[] args) {
        System.out.println("Result: " + nestedRecursion(95));
    }
}
```

Output:

```
Result: 91
```


Write a program to print the GCD of two numbers using recursion.

Source Code:

```
import java.util.Scanner;
public class GCDUsingRecursion {
    public static int GCD(int a,int b){
        if(b==0)
        {
            return a;
        }
        return GCD(b,a%b);
    }
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        System.out.print("Enter the first number:");
        int num1=sc.nextInt();
        System.out.print("Enter the second number:");
        int num2=sc.nextInt();
        int result=GCD(num1,num2);
        System.out.println("GCD:"+result);
    }
}
```

Output:

```
Enter the first number:16
Enter the second number:20
GCD:4
```


Write a program to print the procedure of towerofHanoi for a given number.

Source Code:

```
import java.util.Scanner;
public class TOH {
    public void towerofHanoi(int n,char from_rod,char to_rod,char aux_rod){
        if(n==0)
            return;
        towerofHanoi(n-1,from_rod,aux_rod,to_rod);
        System.out.println("Move disk "+n+" from rod "+from_rod+" to rod "+to_rod);
        towerofHanoi(n-1,aux_rod,to_rod,from_rod);
    }
}
class TOHDemo{
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        System.out.print("Enter the number of disks:");
        int n=sc.nextInt();
        TOH t=new TOH();
        if(n==0)
            System.out.println("There aren't any disks to move");
        else
            t.towerofHanoi(n,'A','B','C');
    }
}
```

Output:

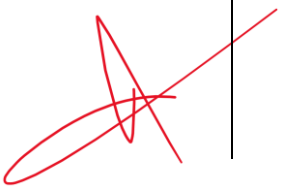
```
Enter the number of disks:2
Move disk 1 from rod A to rod C
Move disk 2 from rod A to rod B
Move disk 1 from rod C to rod B
```

Write a program to print the following pattern using recursion.

1
1 2
1 2 3
1 2 3 4
1 2 3 4 5

Source Code:

```
public class PatternUsingRecursion {
    public void PrintPattern(int n) {
        if (n == 1)
            System.out.print(1);
        else
        {
            PrintPattern(n - 1);
            System.out.print(" " + n);
        }
    }
    public void PrintPattern1(int n) {
        if (n == 1)
        {
            PrintPattern(n);
            System.out.println();
        }
        else
        {
            PrintPattern1(n - 1);
            PrintPattern(n);
            System.out.println();
        }
    }
}
class PatternDemo {
```

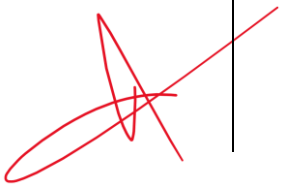
```
    public static void main(String[] args) {
        PatternUsingRecursion p = new PatternUsingRecursion();
        p.PrintPattern1(5);
    }
}
```


Write a program to show all the operations in Binary Search Tree.

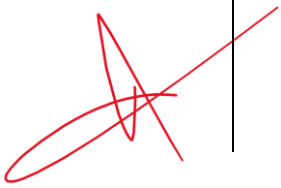
Source Code:

```
import java.util.Scanner;
import java.util.LinkedList;
class Queue {
    LinkedList<Node> que = new LinkedList<>();
    public void enqueue(Node el) {
        que.addFirst(el);
    }
    public Node dequeue() {
        return que.removeLast();
    }
    public boolean isEmpty() {
        return que.isEmpty();
    }
}
class Node {
    public int key;
    Node left, right;
    public Node() {
        left = right = null;
    }
    public Node(int el) {
        this(el, null, null);
    }
    public Node(int el, Node lt, Node rt) {
        key = el;
        left = lt;
        right = rt;
    }
}
```

```
class BST {
    public Node root;
    public BST() {
        root = null;
    }
    public void visit(Node p) {
        System.out.print(p.key + " ");
    }
    public void insert(int el) {
        Node p = root, prev = null;
        while (p != null) {
            prev = p;
            if (p.key < el) {
                p = p.right;
            } else {
                p = p.left;
            }
        }
        if (root == null) {
            root = new Node(el);
        } else if (prev.key < el) {
            prev.right = new Node(el);
        } else {
            prev.left = new Node(el);
        }
    }
    public void inOrder(Node p) {
        if (p != null) {
            inOrder(p.left);
            visit(p);
            inOrder(p.right);
        }
```

```
    }
    public void preOrder(Node p) {
        if (p != null) {
            visit(p);
            preOrder(p.left);
            preOrder(p.right);
        }
    }
    public void postOrder(Node p) {
        if (p != null) {
            postOrder(p.left);
            postOrder(p.right);
            visit(p);
        }
    }
    public void deleteByCopying(int el) {
        Node node, p = root, prev = null;
        while (p != null && p.key != el) {
            prev = p;
            if (p.key < el) {
                p = p.right;
            } else {
                p = p.left;
            }
        }
        node = p;
        if (p != null && p.key == el) {
            if (node.right == null) {
                node = node.left;
            } else if (node.left == null) {
                node = node.right;
            } else {
```

```
            Node tmp = node.left;
            Node previous = node;
            while (tmp.right != null) {
                previous = tmp;
                tmp = tmp.right;
            }
            node.key = tmp.key;
            if (previous == node) {
                previous.left = tmp.left;
            } else {
                previous.right = tmp.right;
            }
        }
        if (p == root) {
            root = node;
        } else if (prev.left == p) {
            prev.left = node;
        } else {
            prev.right = node;
        }
    } else if (root != null) {
        System.out.println("key " + el + " is not in the tree");
    } else {
        System.out.println("tree is empty");
    }
}
void deletebyMerging(int key) {
    root = deletebyMerg(root, key);
}
Node deletebyMerg(Node root, int key) {
    if (root == null) {
        return root;
```

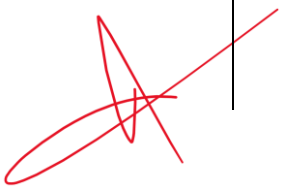
```
    }
    if (key < root.key) {
        root.left = deletebyMerg(root.left, key);
    } else if (key > root.key) {
        root.right = deletebyMerg(root.right, key);
    } else {
        // node with only one child or no child
        if (root.left == null) {
            return root.right;
        } else if (root.right == null) {
            return root.left;
        }
        // node with two children: merge the left subtree with the right subtree
        root.key = findRightmost(root.left).key;
        root.left = deletebyMerg(root.left, root.key);
    }
    return root;
}
Node findRightmost(Node node) {
    while (node.right != null) {
        node = node.right;
    }
    return node;
}
public void breadthFirst() {
    Node p = root;
    Queue queue = new Queue();
    if (p != null) {
        queue.enqueue(p);
        while (!queue.isEmpty()) {
            p = (Node) queue.dequeue();
            visit(p);
```

```
            if (p.left != null) {
                queue.enqueue(p.left);
            }
            if (p.right != null) {
                queue.enqueue(p.right);
            }
        }
    }
  }
}
class QueueDemo {
  public static void main(String[] args) {
    BST tree = new BST();
    Scanner sc = new Scanner(System.in);
    System.out.println("1. Insert an element\n2. PreOrder Traversal\n3. InOrder Traversal\n4. PostOrder Traversal\n5. BreadthFirst Traversal\n6. Delete an element by copying\n7. Delete an element by merging\n8. Exit");
    while (true) {
      System.out.print("\nEnter your choice:");
      int choice = sc.nextInt();
      switch (choice) {
        case 1:
          System.out.print("Enter an element:");
          int el = sc.nextInt();
          tree.insert(el);
          break;
        case 2:
          System.out.println("Preorder traversal:");
          tree.preOrder(tree.root);
          break;
        case 3:
          System.out.println("Inorder traversal:");
          tree.inOrder(tree.root);
```

```
                break;
            case 4:
                System.out.println("Postorder traversal:");
                tree.postOrder(tree.root);
                break;
            case 5:
                System.out.println("Breadth Traversal:\n\n");
                tree.breadthFirst();
                break;
            case 6:
                System.out.print("Enter an element to delete:");
                int el1 = sc.nextInt();
                tree.deleteByCopying(el1);
                break;
            case 7:
                System.out.print("Enter an element to delete:");
                int el2 = sc.nextInt();
                tree.deletebyMerging(el2);
                break;
            case 8:
                System.out.println("Exit Successfully");
                return;
        }
    }
  }
}
```


Output:

```
1. Insert an element
2. PreOrder Traversal
3. InOrder Traversal
4. PostOrder Traversal
5. BreadthFirst Traversal
6. Delete an element by copying
7. Delete an element by merging
8. Exit


Enter your choice:1
Enter an element:50

Enter your choice:1
Enter an element:30

Enter your choice:1
Enter an element:70

Enter your choice:1
Enter an element:20

Enter your choice:1
Enter an element:40

Enter your choice:1
Enter an element:60

Enter your choice:1
Enter an element:80
```



```
Enter your choice:2
Preorder traversal:
50 30 20 40 70 60 80
Enter your choice:3
Inorder traversal:
20 30 40 50 60 70 80
Enter your choice:4
Postorder traversal:
20 40 30 60 80 70 50
Enter your choice:5
Breadth Traversal:


50 30 70 20 40 60 80
Enter your choice:6
Enter an element to delete:20

Enter your choice:3
Inorder traversal:
30 40 50 60 70 80
Enter your choice:7
Enter an element to delete:40

Enter your choice:3
Inorder traversal:
30 50 60 70 80
Enter your choice:8
Exit Successfully
```


Write a program to demonstrate selection sort.

Source Code:

```
import java.util.Scanner;
public class SelectionSortDemo {
    public static void selectionSort(int[] array) {
        int size = array.length;
        // Loop through each element
        for (int step = 0; step < size - 1; step++) {
            int minIndex = step;
            // Find the minimum element in the remaining array
            for (int i = step + 1; i < size; i++) {
                if (array[i] < array[minIndex]) { // Change < to > for descending order
                    minIndex = i;
                }
            }
            // Swap the found minimum element with the first element of the unsorted part
            int temp = array[step];
            array[step] = array[minIndex];
            array[minIndex] = temp;
        }
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        // Taking input for array size
        System.out.print("Enter the number of elements: ");
        int n = sc.nextInt();
        int[] arr = new int[n];
        // Taking input for array elements
        System.out.println("Enter the elements:");
        for (int i = 0; i < n; i++) {
            arr[i] = sc.nextInt();
```

```
        }
        // Sorting the array using Selection Sort
        selectionSort(arr);
        // Display sorted array
        System.out.println("Sorted array:");
        for(int i=0;i<arr.length;i++){
            System.out.print(arr[i]+" ");
        }
    }
}
```

Output:

```
Enter the number of elements: 5
Enter the elements:
3 8 1 6 2
Sorted array:
1 2 3 6 8
```


Write a program to demonstrate Bubble Sort.

Source Code:

```
import java.util.Scanner;
public class BubbleSort {
   // Function to perform Bubble Sort
   public static void bubbleSort(int[] A) {
      int n = A.length;
      for (int i = 0; i < n - 1; i++) {
         for (int j = 0; j < n - i - 1; j++) {
            if (A[j] > A[j + 1]) {
               int temp = A[j];
               A[j] = A[j + 1];
               A[j + 1] = temp;
            }
         }
      }
   }
   // Function to print the array
   public static void printArray(int[] A) {
      System.out.print("Sorted array is: ");
      for (int i = 0; i < A.length; i++) {
         System.out.print(A[i] + " ");
      }
      System.out.println();
   }
   // Main method to test the bubbleSort function
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      System.out.print("Enter the number of elements in the array: ");
      int n = sc.nextInt();
      int[] arr = new int[n];
```

```
        System.out.println("Enter the elements of the array:");
        for (int i = 0; i < arr.length; i++) {
            arr[i] = sc.nextInt();
        }
        bubbleSort(arr);
        printArray(arr);
    }
}
```

Output:

```
Enter the number of elements in the array: 4
Enter the elements of the array:
1 4 2 3
Sorted array is: 1 2 3 4
```


Write a program to demonstrate Radix Sort.

Source Code:

```
import java.util.LinkedList;
import java.util.Scanner;
class Queuee {
    LinkedList q = new LinkedList();
    public void enqueue(Object el) {
// q.addFirst(el);
        q.addLast(el);
    }
    public Object dequeue() {
        return q.removeFirst();
    }
    public boolean isempty() {
        return q.isEmpty();
    }
}
class Radixsort {
    public static int getL(Integer a[]) {
        int len = a[0].toString().length();
        for (Integer v : a) {
            int alen = v.toString().length();
            if (len < alen) {
                len = alen;
            }
        }
        return len;
    }
    public static int getDigPlace(Integer digit, int place) {
        int rem = 0;
        for (int i = 1; i <= place; i++) {
```

```
            rem = digit % 10;
            digit /= 10;
        }
        return rem;
    }
    public static void main(String[] args) {
        Integer arr[] = {1, 2123, 124, 3, 1245, 123, 1428, 38, 237, 45645};
        int pass = getL(arr);
        Queuee[] piles = new Queuee[10];
        for (int i = 0; i < piles.length; i++) {
            piles[i] = new Queuee();
        }
        for (int i = 1; i <= pass; i++) {
            for (int j = 0; j < arr.length; j++) {
                Integer no = arr[j];
                Integer pilepos = getDigPlace(no, i);
                piles[pilepos].enqueue(no);
            }//end of pass1
//Arraging the elements of piles to array again
            int l = 0;
            for (int j = 0; j < piles.length; j++) {
                while (!piles[j].isempty()) {
                    arr[l] = (Integer) piles[j].dequeue();
                    l++;
                }
            }
            System.out.println("After pass:" + (i));
            for (int v : arr) {
                System.out.print(v + "\t");
            }
            System.out.println("");
        }
```

```
    }
}
```

Output:

```
After pass:1
1       2123    3       123     124     1245    45645   237     1428    38
After pass:2
1       3       2123    123     124     1428    237     38      1245    45645
After pass:3
1       3       38      2123    123     124     237     1245    1428    45645
After pass:4
1       3       38      123     124     237     1245    1428    2123    45645
After pass:5
1       3       38      123     124     237     1245    1428    2123    45645
```


Write a program to demonstrate Insertion Sort.

Source Code:

```
import java.util.Scanner;
public class InsertionSort {
   // Method to perform Insertion Sort
   public static void insertion_sort(int A[], int n) {
      for (int i = 0; i < n; i++) {
         int temp = A[i];
         int j = i;
         while (j > 0 && temp < A[j - 1]) {
            A[j] = A[j - 1];
            j = j - 1;
         }
         A[j] = temp;
      }
   }
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      System.out.print("Enter the number of elements: ");
      int n = sc.nextInt();
      int[] arr = new int[n];
      System.out.println("Enter " + n + " elements:");
      for (int i = 0; i < n; i++) {
         arr[i] = sc.nextInt();
      }
      insertion_sort(arr, n);
      // Display sorted array
      System.out.println("Sorted Array:");
      for(int i=0;i<arr.length;i++){
         System.out.print(arr[i]+" ");
      }
```

```
    }
}
```

Output:

```
Enter the number of elements: 5
Enter 5 elements:
3 1 2 5 4
Sorted Array:
1 2 3 4 5
```


Write a program to demonstrate Heap Sort.

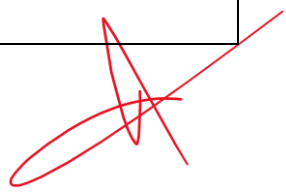
Source Code:

```
import java.util.Scanner;
public class HeapSort {
    static void heapify(int arr[], int n, int i) {
        int largest = i;  // Initialize largest as root
        int l = 2 * i + 1;  // left = 2*i + 1
        int r = 2 * i + 2;  // right = 2*i + 2
        // If left child is larger than root
        if (l < n && arr[l] > arr[largest]) {
            largest = l;
        }
        // If right child is larger than largest so far
        if (r < n && arr[r] > arr[largest]) {
            largest = r;
        }
        // If largest is not root
        if (largest != i) {
            swap(arr, i, largest);
            // Recursively heapify the affected sub-tree
            heapify(arr, n, largest);
        }
    }
    static void heapSort(int arr[], int n) {
        // Build heap (rearrange array)
        for (int i = n / 2 - 1; i >= 0; i--) {
            heapify(arr, n, i);
        }
        // One by one extract an element from heap
        for (int i = n - 1; i > 0; i--) {
            // Move current root to end
```

```
            swap(arr, 0, i);
            // Call max heapify on the reduced heap
            heapify(arr, i, 0);
        }
    }
    static void swap(int[] arr, int a, int b) {
        int temp = arr[a];
        arr[a] = arr[b];
        arr[b] = temp;
    }
    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the number of elements: ");
        int n = sc.nextInt();
        int arr[] = new int[n];
        System.out.println("Enter the elements:");
        for (int i = 0; i < n; i++) {
            arr[i] = sc.nextInt();
        }
        heapSort(arr, n);
        for(int i=0;i<arr.length;i++){
            System.out.print(arr[i]+" ");
        }
    }
}
```

Output:

```
Enter the number of elements: 5
Enter the elements:
5 2 4 1 3
1 2 3 4 5
```



Write a program to demonstrate Quick Sort.

Source Code:

```
import java.util.Scanner;
public class QuickSort {
   // Partition function
   static int partition(int[] arr, int low, int high) {
      int pivot = arr[high];
      int i = low - 1;
      for (int j = low; j <= high - 1; j++) {
         if (arr[j] < pivot) {
            i++;
            swap(arr, i, j);
         }
      }
      swap(arr, i + 1, high);
      return i + 1;
   }
   // Swap function
   static void swap(int[] arr, int i, int j) {
      int temp = arr[i];
      arr[i] = arr[j];
      arr[j] = temp;
   }
   // QuickSort function
   static void quickSort(int[] arr, int low, int high) {
      if (low < high) {
         int pi = partition(arr, low, high);
         quickSort(arr, low, pi - 1);
         quickSort(arr, pi + 1, high);
      }
   }
```

```
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the number of elements: ");
        int n = sc.nextInt();
        int[] arr = new int[n];
        System.out.println("Enter the elements: ");
        for (int i = 0; i < n; i++) {
            arr[i] = sc.nextInt();
        }
        quickSort(arr, 0, n - 1);
        System.out.println("Sorted array: ");
        for (int val : arr) {
            System.out.print(val + " ");
        }
    }
}
```

Output:

```
Enter the number of elements: 5
Enter the elements:
1 2 6 4 5
Sorted array:
1 2 4 5 6
```


Write a program to demonstrate Merge Sort.

Source Code:

```
import java.util.Scanner;
public class MergeSort {
    public static void mergeSort(int[] array) {
        int n = array.length;
        if (n < 2) {
            return;
        }
        int mid = n / 2;
        int[] left = new int[mid];
        int[] right = new int[n - mid];
        // Copying elements to left and right subarrays
        for (int i = 0; i < mid; i++) {
            left[i] = array[i];
        }
        for (int i = mid; i < n; i++) {
            right[i - mid] = array[i];
        }
        mergeSort(left);
        mergeSort(right);
        merge(left, right, array);
    }
    // Merging two sorted arrays
    public static void merge(int[] left, int[] right, int[] array) {
        int i = 0, j = 0, k = 0;
        while (i < left.length && j < right.length) {
            if (left[i] <= right[j]) {
                array[k++] = left[i++];
            } else {
                array[k++] = right[j++];
```

```
            }
        }
        while (i < left.length) {
            array[k++] = left[i++];
        }
        while (j < right.length) {
            array[k++] = right[j++];
        }
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the number of elements: ");
        int n = sc.nextInt();
        int[] array = new int[n];
        System.out.println("Enter the elements:");
        for (int i = 0; i < n; i++) {
            array[i] = sc.nextInt();
        }
        mergeSort(array);
        System.out.println("Sorted array:");
        for (int i = 0; i < array.length; i++) {
            System.out.print(array[i] + " ");
        }
    }
}
```

Output:

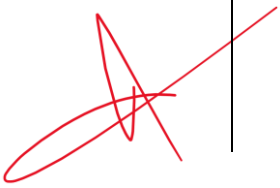
```
Enter the number of elements: 5
Enter the elements:
1 2 5 3 4
Sorted array:
1 2 3 4 5
```

Write a program for hashing using linear probing.

Source Code:

```
public class Hashing {
    int tablesize;
    Integer[] arr;
    public Hashing(int tablesize) {
        this.tablesize = tablesize;
        arr = new Integer[tablesize];
    }
    public int hashfunction(int key) {
        return key % this.tablesize;
    }
    public boolean collision(int index) {
        return (arr[index] != null);
    }
    public void inserthash(int key) {
        int index = hashfunction(key);
        int i = 1;
        while (collision(index)) {
            index = (hashfunction(key) + i) % this.tablesize;
            i++;
        }
        arr[index] = key;
    }
    public void search(int key) {
        boolean found = false;
        int index = hashfunction(key);
        int i = 1;
        if (arr[index] == key) {
            found = true;
        } else {
```

```
        while (arr[index] != key && i <= arr.length) {
            index = (hashfunction(key) + i) % this.tablesize;
            if (arr[index] == key) {
                found = true;
                break;
            }
            i++;

        }
    }
    if (found) {
        System.out.println(key + "is found in table");
    } else {
        System.out.println("not found");
    }
}
void deletehash(int key) {
    boolean found = false;
    int index = hashfunction(key);
    int i = 1;
    if (arr[index] == key) {
        found = true;
    } else {
        while (arr[index] != key && i <= arr.length) {
            index = (hashfunction(key) + i) % this.tablesize;
            if (arr[index] == key) {
                found = true;
                break;
            }
            i++;
        }
    }
```

```
        if (found) {
            int keytodelete = arr[index];
            arr[index] = null;
            System.out.println(keytodelete + "deleted from table");
        } else {
            System.out.println("no such element");
        }
    }
    public void printAll() {
        for (Integer var : arr) {
            System.out.println(var);
        }
    }
}
class Hashdemo {
    public static void main(String args[]) {
        int[] input = {5, 7, 12, 14, 3};
        Hashing h = new Hashing(input.length);
        for (int var : input) {
            h.inserthash(var);
        }
        h.printAll();
        h.search(7);
        h.search(14);
        h.search(2);
        h.deletehash(7);
        h.printAll();
    }
}
```

Output:

```
5
3
7
12
14
7is found in table
14is found in table
not found
7deleted from table
5
3
null
12
14
```


Write a program to represent graph using an adjacency matrix.

Source Code:

```
import java.util.Scanner;
class GraphMatrix {
    int[][] adjacencyMatrix;
    int vertices;
    // Constructor
    public GraphMatrix(int vertices) {
        this.vertices = vertices;
        adjacencyMatrix = new int[vertices][vertices];
    }
    // Method to add an edge to the graph
    public void addEdge(int source, int destination) {
        if (source >= 0 && source < vertices && destination >= 0 && destination < vertices) {
            adjacencyMatrix[source][destination] = 1;
            adjacencyMatrix[destination][source] = 1; // For undirected graph
        } else {
            System.out.println("Invalid edge!");
        }
    }
    // Method to display the adjacency matrix
    public void displayMatrix() {
        System.out.println("Adjacency Matrix:");
        for (int i = 0; i < vertices; i++) {
            for (int j = 0; j < vertices; j++) {
                System.out.print(adjacencyMatrix[i][j] + " ");
            }
            System.out.println();
        }
    }
```

```
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the number of vertices: ");
        int vertices = sc.nextInt();
        GraphMatrix graph = new GraphMatrix(vertices);
        System.out.print("Enter the number of edges: ");
        int edges = sc.nextInt();
        System.out.println("Enter the edges (source destination):");
        for (int i = 0; i < edges; i++) {
            int source = sc.nextInt();
            int destination = sc.nextInt();
            graph.addEdge(source, destination);
        }
        graph.displayMatrix();
    }
}
```

Output:

```
Enter the number of vertices: 4
Enter the number of edges: 4
Enter the edges (source destination):
0 1
0 2
1 2
2 3
Adjacency Matrix:
0 1 1 0
1 0 1 0
1 1 0 1
0 0 1 0
```


Write a program to add and remove edges from a graph.

Source Code:

```
import java.util.Scanner;
class Graph {
    int[][] adjacencyMatrix;
    int vertices;
    public Graph(int vertices) {
        this.vertices = vertices;
        adjacencyMatrix = new int[vertices][vertices];
    }
    public void addEdge(int source, int destination) {
        if (isValidVertex(source) && isValidVertex(destination)) {
            adjacencyMatrix[source][destination] = 1;
            adjacencyMatrix[destination][source] = 1;
            System.out.println("Edge added between " + source + " and " + destination);
        } else {
            System.out.println("Invalid edge!");
        }
    }
    public void removeEdge(int source, int destination) {
        if (isValidVertex(source) && isValidVertex(destination) && adjacencyMatrix[source][destination] == 1) {
            adjacencyMatrix[source][destination] = 0;
            adjacencyMatrix[destination][source] = 0;
            System.out.println("Edge removed between " + source + " and " + destination);
        } else {
            System.out.println("No such edge exists!");
        }
    }
    private boolean isValidVertex(int v) {
        return v >= 0 && v < vertices;
    }
```

```
public void displayMatrix() {
    System.out.println("Adjacency Matrix:");
    for (int i = 0; i < vertices; i++) {
        for (int j = 0; j < vertices; j++) {
            System.out.print(adjacencyMatrix[i][j] + " ");
        }
        System.out.println();
    }
}
public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    System.out.print("Enter the number of vertices: ");
    int vertices = sc.nextInt();
    Graph graph = new Graph(vertices);
    System.out.println("\nMenu:");
    System.out.println("1. Add Edge");
    System.out.println("2. Remove Edge");
    System.out.println("3. Display Graph");
    System.out.println("4. Exit");
    while (true) {
        System.out.print("Enter your choice: ");
        int choice = sc.nextInt();
        switch (choice) {
            case 1:
                System.out.print("Enter source and destination to add edge: ");
                int srcAdd = sc.nextInt();
                int destAdd = sc.nextInt();
                graph.addEdge(srcAdd, destAdd);
                break;
            case 2:
                System.out.print("Enter source and destination to remove edge: ");
                int srcRemove = sc.nextInt();
```

```
                int destRemove = sc.nextInt();
                graph.removeEdge(srcRemove, destRemove);
                break;
            case 3:
                graph.displayMatrix();
                break;
            case 4:
                System.out.println("Exit Successfully");
                return;
            default:
                System.out.println("Invalid choice!");
        }
    }
}
}
```

Output:

```
Enter the number of vertices: 4


Menu:
1. Add Edge
2. Remove Edge
3. Display Graph
4. Exit
Enter your choice: 1
Enter source and destination to add edge: 0 1
Edge added between 0 and 1
Enter your choice: 1
Enter source and destination to add edge: 1 2
Edge added between 1 and 2
Enter your choice: 3
Adjacency Matrix:
```

```
0 1 0 0
1 0 1 0
0 1 0 0
0 0 0 0
Enter your choice: 2
Enter source and destination to remove edge: 1 2
Edge removed between 1 and 2

Enter your choice: 3
Adjacency Matrix:
0 1 0 0
1 0 0 0
0 0 0 0
0 0 0 0
Enter your choice: 4
Exit Successfully
```