

Array In Java

An array is an indexed collection of fixed number of homogeneous data elements. The main advantage of arrays is we can represent huge number of values by using single variable. So that readability of the code will be improved. But the main disadvantage of array is fixed in size that is one's we create an array there is no chance of increasing or decreasing in the size based on our requirement. Hence to use arrays concept compulsory we should know the size in advance, which may not possible always.

Array Declaration:

One-Dimensional, Two-Dimensional & Three-Dimensional Array Declaration:

```
class test
{
public static void main(String args[])
{
int [] x; //recommended declaration.
int []a;
int b[];
}
}
```

```
class test
{
public static void main(String args[])
{
int [][] x;
int [][]a;
int b[][];
int[] []c;
int[] d[];
int []r[];
}
}
```

```
class test
{
public static void main(String args[])
{
int[][][] a;
int [][][]b;
int c[][][];
int[] [][]d;
int[] []e[];
int[][] []f;
int[][] g[];
int [][]h[];
int []i[][];
}
}
```

Recommended because name is clearly separated from type. At the time of declaration we can't specify the size we will get compile time error.

int[6] x; => invalid statement

int[] x => valid statement;

Q. which of the following are valid statement:

```
class test
{
public static void main(String args[])
{
int[] a,b; //a = 1, b = 1

int[] c[],d; // c = 2, d = 1

int[] g[],h[]; // g = 2, h = 2

int[] []e,f; // e = 2, f = 2

int[] []i,j[]; // i = 2, j = 3

int[] []s, []t; //C.E
}
}
```

If we want to specify dimension before the variable. That facility is applicable only for first variable in a declaration. If we are trying to apply for remaining variables we will get compile time error. `int[] []a, []b, []c` ; it is an compile time error. because of `[]b, []c`;

Array Creation:

Every array in java is an object only. Hence we can creates array. by using new operator.

```
class test
{
public static void main(String args[])
{
int[] a = new int[3];
}
}
```

For every array type corresponding classes are available and these classes are part of java language and not available to the program level. `[]` = 1-d array , `[][]` = 2-d array

`int[]` = `[]`

`int[][]` = `[][]`

`double[]` = `[]`

`short[]` = `[]`

`byte[]` = `[]`

`boolean[]` = `[]`

```
class test
{
public static void main(String args[])
{
int[][] a = new int[3][2];
System.out.println(a.getClass().getName());
}
}
```

```
C:\Users\Atish kumar sahu\desktop>java test
[[I

C:\Users\Atish kumar sahu\desktop>javac test.java

C:\Users\Atish kumar sahu\desktop>java test
[[I
```

Case Studies In Array:

case 01:

At the time of array creation compulsory we should specify the size. Otherwise we will get compile time error.

```
class test
{
public static void main(String args[])
{
int[] a = new int[]; //CE
int[] v = new int[10]; //correct
}
}
```

case 02:

It is legal to have an array with size zero(0) in java. `int[] x = new int[0];`

case 03:

If we are trying to specify array size with some negative int value then we will get run time exception saying "NegativeArraySizeException". `Int[] n = new int[-6]` -> Runtime error.

case 04:

The maximum allowed array size in java is 2147483647 which is the maximum value of int data type. `int[] x = new int[2147483647]` -> valid but because of lack of heap memory there is an exception. `int[] x = new int[2147483648]` -> invalid.

case 05:

To specify array size the allowed data types are byte, short, char, int. if we are trying to specify any other type then we will get compile time error.

```
class test
{
    public static void main(String args[])
    {
        int[] c = new int[10]; //valid
        int[] g = new int['a']; //valid

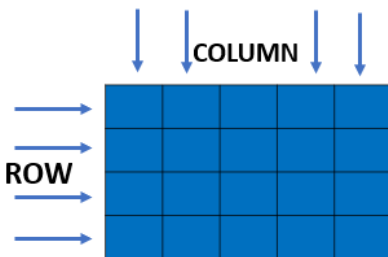
        byte b = 0;
        int[] x = new int[b]; //valid

        short s = 10;
        int[] o = new int[s]; //valid

        int[] x = new int[55l]; // CE
    }
}
```

Two-Dimensional Array:

In java 2-dimensional array is not implemented by matrix style. Sun people followed array of arrays approach for multi-dimensional array creation. The main advantage of this approach is memory utilization will be improved.



```
class test
{
    public static void main(String args[])
    {
        int[][] x = new int[2][];
        x[0] = new int[2];
        x[1] = new int[3];
    }
}
```

```
class test
{
    public static void main(String args[])
    {
        int[][][] x = new int[2][][];
        x[0] = new int[3][];
        x[0][0] = new int[1];
        x[0][1] = new int[2];
        x[0][2] = new int[3];

        x[1] = new int [2][2];
    }
}
```

Q. Which Of The Following Are Valid Array Creation?

```
class test
{
public static void main(String args[])
{
int[] a = new int[]; // invalid
int[] b = new int[3]; //valid
int[][] c = new int[][]; //invalid
int[][] d = new int[3][]; //valid
int[][] e = new int[][4]; //invalid
int[][] f = new int[3][4]; //valid
int[][][] g = new int[3][4][5]; //valid
int[][][] h = new int[3][4][]; //valid
int[][][] i = new int[3][][5]; //invalid
int[][][] j = new int[][4][5]; //invalid
}
}
```

Array Initialization:

Once we create an array every element by default initialized with default values. Whenever we are trying to print any reference variable internally to String method will be called. Which is implemented by default to return the String in the following form "classname@hashcode_in_hexadecimal_form".

```
class test
{
public static void main(String args[])
{
int[] x = new int[3];
System.out.println(x);
System.out.println(x[0]);
System.out.println(x[1]);
System.out.println(x[2]);
}
}
```

```
C:\Users\Atish kumar sahu\desktop>javac test.java
```

```
C:\Users\Atish kumar sahu\desktop>java test
```

```
[I@15db9742
```

```
0
```

```
0
```

```
0
```

```
class test
{
public static void main(String args[])
{
int[][] x = new int[3][2];
System.out.println(x);
System.out.println(x[0]);
System.out.println(x[0][0]);
}
}
```

```
C:\Users\Atish kumar sahu\desktop>javac test.java
```

```
C:\Users\Atish kumar sahu\desktop>java test
[[I@15db9742
[I@6d06d69c
0
```

```
class test
{
public static void main(String args[])
{
int[][] x = new int[3][];
System.out.println(x);
System.out.println(x[0]);
System.out.println(x[0][0]);
}
}
```

```
C:\Users\Atish kumar sahu\desktop>javac test.java
```

```
C:\Users\Atish kumar sahu\desktop>java test
[[I@15db9742
null
Exception in thread "main" java.lang.NullPointerException
    at test.main(test.java:8)
```

If we are trying to perform any operation on null then we will get runtime exception saying "NullPointerException".

Once we create an array every array element by default is initialized with default values. If we are not satisfied by default values. Then we can override these values with our customized values.

```

class test
{
public static void main(String args[])
{
int[] x = new int[4];
x[0] = 10;
x[1] = 20;
x[2] = 30;
x[3] = 40;
System.out.println(x[0]+"--"+x[1]+"--"+x[2]+"--"+x[3]);
}
}

```

```

C:\Users\Atish kumar sahu\desktop>javac test.java

```

```

C:\Users\Atish kumar sahu\desktop>java test
10--20--30--40

```

```

class test
{
public static void main(String args[])
{
int[] x = new int[4];
x[0] = 10;
x[1] = 20;
x[2] = 30;
x[3] = 40;
x[4] = 50;
}
}

```

```

C:\Users\Atish kumar sahu\desktop>javac test.java

```

```

C:\Users\Atish kumar sahu\desktop>java test
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException
: 4
    at test.main(test.java:10)

```

X[-4] = 50; -> java.lang.ArrayIndexOutOfBoundsException

X[2.5] = 90; -> Compile Time Error.

If we are trying to access array element with out of range index(either +ve or -ve) value. then we will get runtime exception saying "java.lang.ArrayIndexOutOfBoundsException".

Array Declaration, Creation Initialization In A Single Line:

We can declare, create and initialize an array in a single line, like shortcut representation. We can extend this representation for multidimensional arrays also. If we want to use this shortcut compulsory we should perform all activities in a single line. If we are trying to divide into multiple line then we will get compile time error.

```
class test
{
public static void main(String args[])
{
int[] x ;
x = new int[4];
x[0] = 10;
x[1] = 20;
x[2] = 30;
x[3] = 40;
x[4] = 50;
```

```
int[] a = {10, 20, 30};
char[] b = {'a', 'b', 'c'};
String[] s = {"a", "aa", "aaa"};
}
}
```

```
class test
{
public static void main(String args[])
{
int[][] x = {{10,20},{30,40,50}};
}
}
```

```
class test
{
public static void main(String args[])
{
int[][][] x = {{{10,20,30},{40,50,60}},{70,80},{90,100,110}}};
System.out.println(x[0][1][2]); //60
System.out.println(x[1][0][1]); //80
System.out.println(x[2][0][0]); //RE
System.out.println(x[1][2][0]); //RE
System.out.println(x[1][1][1]); //100
System.out.println(x[2][1][4]); //RE
}
}
```



```
class test
{
public static void main(String args[])
{
int[] c = {10, 20, 30};
int[] x;
x = {10, 20, 30};
}
}
```

```
C:\Users\Atish kumar sahu\desktop>javac test.java
test.java:7: error: illegal start of expression
x = {10, 20, 30};
    ^
test.java:7: error: not a statement
x = {10, 20, 30};
    ^
test.java:7: error: ';' expected
x = {10, 20, 30};
    ^
test.java:9: error: class, interface, or enum expected
}
^
4 errors
```

length vs length():

length:

Length is a final variable applicable for arrays. Length variable represents the size of the array.

```
class test
{
public static void main(String args[])
{
int[] x = new int[6];
// System.out.println(x.length()); //CE
System.out.println(x.length);
}
}
```

```
C:\Users\Atish kumar sahu\desktop>javac test.java
```

```
C:\Users\Atish kumar sahu\desktop>java test
6
```

length():

Length() method is a final method applicable for String object. length method returns number of character present in the String.

```
class test
{
public static void main(String args[])
{
String s = "durga";
//System.out.println(s.length); //CE
System.out.println(s.length());
}
}
```

```
C:\Users\Atish kumar sahu\desktop>javac test.java
```

```
C:\Users\Atish kumar sahu\desktop>java test
5
```

Length variable is applicable for array but not for String Objects. length() method is applicable for String Objects but not for arrays.

```
class test
{
public static void main(String args[])
{
String[] s = {"a", "aa", "aaa", "aaaa"};
System.out.println(s.length);
//System.out.println(s.length());
//System.out.println(s[0].length);
System.out.println(s[0].length());
}
}
```

```
C:\Users\Atish kumar sahu\desktop>javac test.java
test.java:7: error: cannot find symbol
System.out.println(s.length());
                    ^
    symbol:   method length()
    location: variable s of type String[]
test.java:8: error: cannot find symbol
System.out.println(s[0].length);
                    ^
    symbol:   variable length
    location: class String
2 errors

C:\Users\Atish kumar sahu\desktop>javac test.java

C:\Users\Atish kumar sahu\desktop>java test
4
1
```

In multi-dimensional array length variable represents only base size but not total size. There is no direct way to find total length of multi-dimensional array but indirectly we can find as follows.

```
class test
{
public static void main(String args[])
{
int[][] x = new int[6][3];
System.out.println(x.length);
System.out.println(x[0].length);
}
}
```

```
C:\Users\Atish kumar sahu\desktop>javac test.java
```

```
C:\Users\Atish kumar sahu\desktop>java test
6
3
```

```
class test
{
public static void main(String args[])
{
int[][] x = new int[3][3];
System.out.println(x.length);
System.out.println(x[0].length+ x[1].length+ x[2].length);
}
}
```

```
C:\Users\Atish kumar sahu\desktop>javac test.java
```

```
C:\Users\Atish kumar sahu\desktop>java test
3
9
```

Anonymous Array:

Sometimes we can declare an array without name such type of nameless arrays are called anonymous arrays. The main purpose of anonymous arrays is just for instant use(one time uses). We can create anonymous array as follows “new int[] {10,20,30}”. While creating anonymous arrays we can’t specify the size, otherwise we will get compile time error.

new int[3]{10,20,30} -> invalid

new int[]{20,30,40} - > valid

new int[][] {{10,20},{30,40,50}} we can create multidimensional anonymous arrays also.

Based on our requirement we can give the name for anonymous array then it is no longer anonymous. “Int[] x = new int[] {10,20,30};”

```
class test
{
    public static void main(String args[])
    {
        sum(new int[] {10,20,30,40});
    }

    public static void sum(int[] x)
    {
        int total = 0;
        for(int x1 : x)
        {
            total = total + x1;
        }
        System.out.println(total);
    }
}
```

```
C:\Users\Atish kumar sahu\desktop>javac test.java
```

```
C:\Users\Atish kumar sahu\desktop>java test
100
```

In above program just to call sum method we required an array. but after completing that sum method call we are not using that array anymore. For this one time requirement anonymous array is the best choice.

Array Element Assignment

case 01:

In the case of primitive type array as array element we can provide any type which can be implicitly promoted to declared type.

```
class test
{
public static void main(String args[])
{
int[] x = new int[5];
x[0] = 10;
x[1] = 'a';
byte b = 20;
x[2] = b;
short s = 33;
x[3] = s;
//x[4] = 111l; //CE
}
}
```

case 02:

In case of object type array as array elements we can provide either declared type objects or its child class objects.

```
class test
{
public static void main(String args[])
{
Object[] a = new Object[3];
a[0] = new Object();
a[1] = new String("aaa");
a[2] = new Integer(10);

Number[] b = new Number[7];
b[0] = new Byte((byte)55);
b[1] = new Short((short)124);
b[2] = new Integer(11);
b[3] = new Long(12345l);
b[4] = new Float(11.22f);
b[5] = new Double(147.266d);
b[6] = new String("dddd");//CE -> incompatible types: String cannot be converted to Number
}
}
```

case 03:

For interface type arrays as array elements its implementation class objects are allowed.

```
class test
{
    public static void main(String args[])
    {
        Runnable[] r = new Runnable[10];
        r[0] = new Thread();
    }
}
```

ARRAY TYPE

ELEMENT TYPE

Primitive array ===== any type which can be implicitly promoted to declared type

Object array ===== either declared type or its child class objects

Abstract array ===== its child class objects are allowed

Interface array ===== its implementation objects are allowed

Array Variable Assignment

case 01:

Element level promotion are not applicable at array level. Example: char element can be promoted to int type, whereas char array can't be promoted to int array.

```
class test
{
    public static void main(String args[])
    {
        int[] x = {10, 20, 30};
        char[] y = {'a', 'b', 'c'};

        int[] b = x;
        // int[] c = y; //CE = incompatible types: char[] cannot be converted to int[]
    }
}
```

case 02:

In the case of object type arrays child class type array can be promoted to parent class type array.

```
String[] s = {"aa", "aaa", "a"};  
Object o = s;
```

```
char -> int = valid  
char[] -> int[] = invalid  
  
int -> double = valid  
int[] -> double[] = invalid  
  
float -> int = invalid  
float[] -> int[] = invalid  
  
String -> Object = valid  
String[] -> Object[] = valid
```

case 03:

Whenever we are assigning one array to another array internal elements won't be copied just reference variables will be reassigned.

```
class test  
{  
    public static void main(String args[])  
    {  
        int[] a = {10,20,30,40};  
        int[] b = {50,60,70};  
        System.out.println(a = b);  
        System.out.println(b = a);  
    }  
}
```


case 04:

Whenever we are assigning one array to another array dimension must be matched for example in the place of one-dimensional int array we should provide one dimensional array only. If we are trying to provide any other dimension. Then we will get compile time error.

```
class test
{
    public static void main(String args[])
    {
        int[][] a = new int[3][];
        a[0] = new int[2][3];

        a[0] = 10;

        a[0] = new int[2];
    }
}
```

```
C:\Users\Atish kumar sahu\desktop>javac test.java
test.java:6: error: incompatible types: int[][] cannot be converted
    to int[]
a[0] = new int[2][3];
    ^
test.java:8: error: incompatible types: int cannot be converted to
int[]
a[0] = 10;
    ^
2 errors
```

Whenever we are assigning one array to another array both dimension and types must be matched but sizes are not required to match.

```
class test
{
    public static void main(String args[])
    {
        for(int i = 0; i <= args.length; i++)
        {
            System.out.println(args[i]);
        }
    }
}
```

```
C:\Users\Atish kumar sahu\desktop>javac test.java

C:\Users\Atish kumar sahu\desktop>java test a b c d e
a
b
c
d
e
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException
: 5
    at test.main(test.java:7)
```

```
class test
{
public static void main(String args[])
{
int[][] a = new int[4][3]; //5obj
a[0] = new int[4]; //1obj
a[1] = new int[2]; //2obj
a = new int[3][2]; //4obj
//12 object created
//8 object become garbage collection
}
}
```

Array To String Conversion:

```
import java.util.Arrays;
class test
{
public static void main(String args[])
{
String[] str = {"a", "aa", "aaa"};
System.out.println(str[0]);
System.out.println(str[1]);
System.out.println(str[2]);

String AS = Arrays.toString(str);
System.out.println(AS);
System.out.println(AS.getClass().getSimpleName());
}
}
```

```
C:\Users\Atish kumar sahu\desktop>javac test.java

C:\Users\Atish kumar sahu\desktop>java test
a
aa
aaa
[a, aa, aaa]
String
```

AS.getClass().getSimpleName() is used for to find the data is which data type.

String To Array Conversion:

```
import java.util.Arrays;
class test
{
    public static void main(String args[])
    {
        String s = new String("ATISH KUMAR SAHU");
        System.out.println(s);
        System.out.println(s.getClass().getSimpleName());

        String[] sa1 = s.split("");
        for(String a : sa1)
        {
            System.out.println(a);
        }
    }
}
```

```
C:\Users\Atish kumar sahu\desktop>javac test.java

C:\Users\Atish kumar sahu\desktop>java test
ATISH KUMAR SAHU
String
A
T
I
S
H

K
U
M
A
R

S
A
H
U
```

Array Important Points:

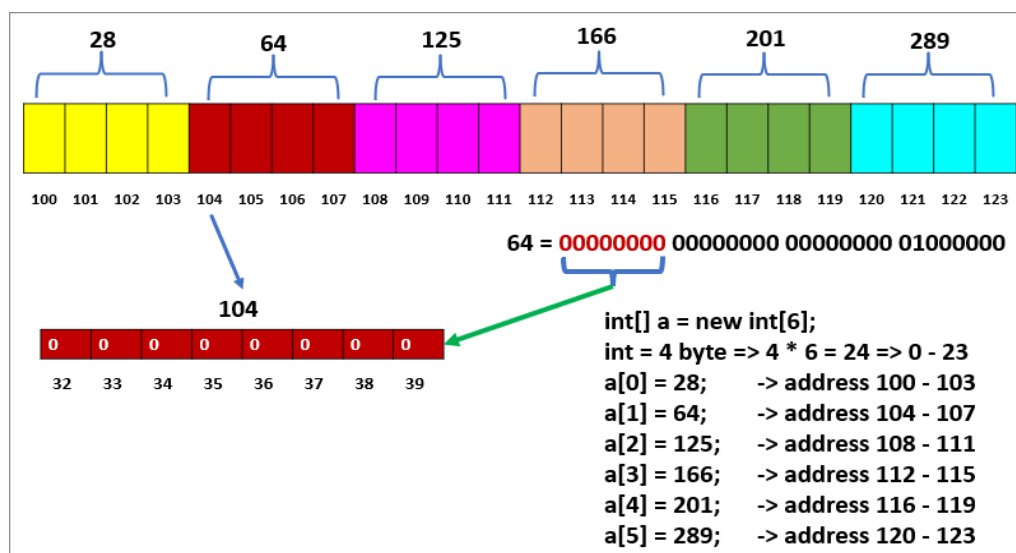
Group of same type variable. Size of array is specified in advance. Each variable in array has an index value starting from 0. Array is a contiguous memory allocation. In array deletion involves movement of element forward and insertion involves movement of elements backward. Merging of array is very difficult and array have fixed size so it follows static or compile time memory allocation.

Advantages Of Array:

1. Arrays are stored in contiguous memory and each element is identified with its index. So elements can be easily accessing elements using the positional value called index. The location of element can be calculated by using base address or address of first element.
2. Array are contiguous so all operations can be performed with high efficiency.
3. Array are easy understand and easy to implement in a programming language.

Disadvantages Of Array:

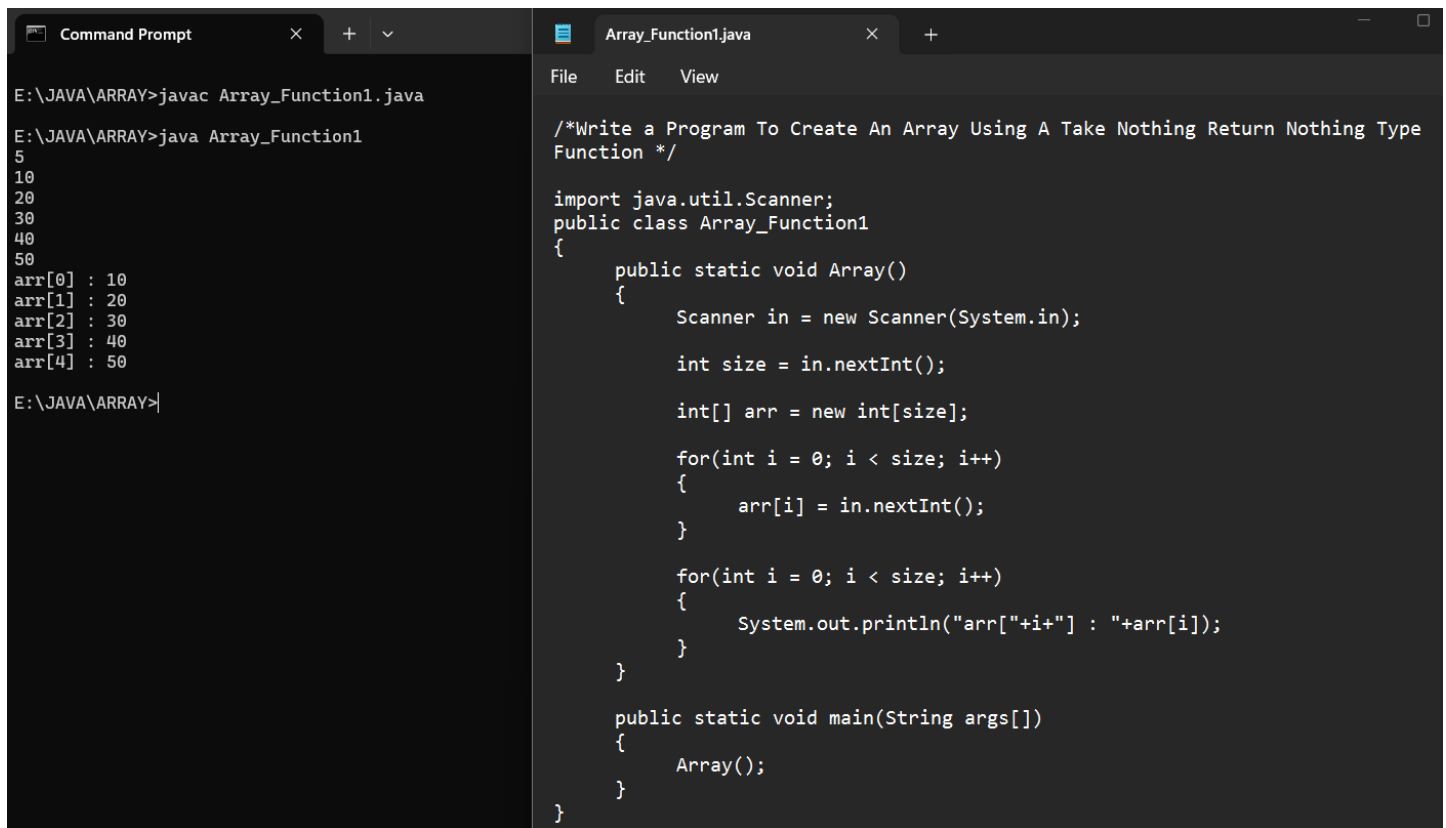
1. Insertion in array needs all elements after the position of insertion to move one place backwards .
2. Deletion in array requires all elements to move one place forward after the deleted element to fill the hole thus created.
3. Array size once defined cannot be increased or decreased (In theory, but implementation may be programming language specific)
4. Array elements must be stored in contiguous memory locations.



The End

Function & Array

Take Nothing Return Nothing Type Function & Array:



```
Command Prompt
E:\JAVA\ARRAY>javac Array_Function1.java
E:\JAVA\ARRAY>java Array_Function1
5
10
20
30
40
50
arr[0] : 10
arr[1] : 20
arr[2] : 30
arr[3] : 40
arr[4] : 50
E:\JAVA\ARRAY>

Array_Function1.java
File Edit View
/*Write a Program To Create An Array Using A Take Nothing Return Nothing Type
Function */

import java.util.Scanner;
public class Array_Function1
{
    public static void Array()
    {
        Scanner in = new Scanner(System.in);

        int size = in.nextInt();

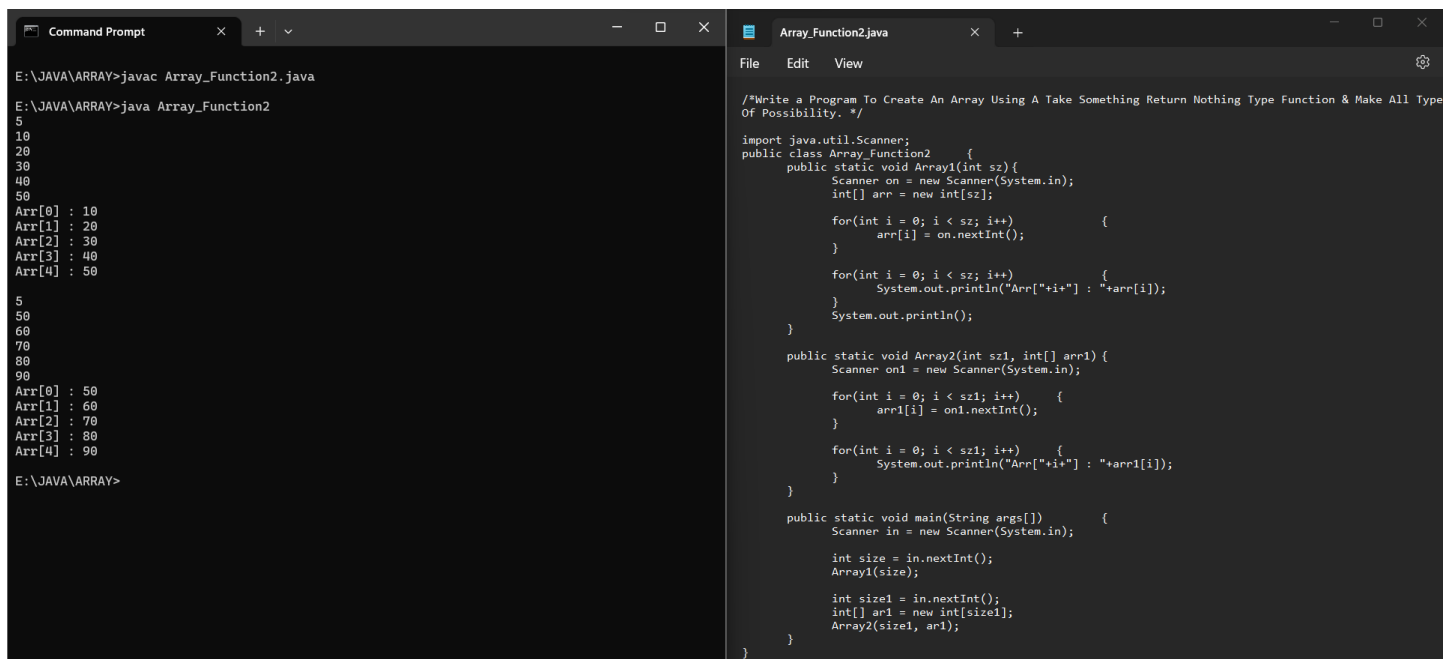
        int[] arr = new int[size];

        for(int i = 0; i < size; i++)
        {
            arr[i] = in.nextInt();
        }

        for(int i = 0; i < size; i++)
        {
            System.out.println("arr["+i+"] : "+arr[i]);
        }
    }

    public static void main(String args[])
    {
        Array();
    }
}
```

Take Nothing Return Something Type Function & Array:



```
Command Prompt
E:\JAVA\ARRAY>javac Array_Function2.java
E:\JAVA\ARRAY>java Array_Function2
5
10
20
30
40
50
Arr[0] : 10
Arr[1] : 20
Arr[2] : 30
Arr[3] : 40
Arr[4] : 50
5
50
60
70
80
90
Arr[0] : 50
Arr[1] : 60
Arr[2] : 70
Arr[3] : 80
Arr[4] : 90
E:\JAVA\ARRAY>

Array_Function2.java
File Edit View
/*Write a Program To Create An Array Using A Take Something Return Nothing Type Function & Make All Type
Of Possibility. */

import java.util.Scanner;
public class Array_Function2
{
    public static void Array1(int sz){
        Scanner on = new Scanner(System.in);
        int[] arr = new int[sz];

        for(int i = 0; i < sz; i++)
        {
            arr[i] = on.nextInt();
        }

        for(int i = 0; i < sz; i++)
        {
            System.out.println("Arr["+i+"] : "+arr[i]);
        }
        System.out.println();
    }

    public static void Array2(int sz1, int[] arr1){
        Scanner on1 = new Scanner(System.in);

        for(int i = 0; i < sz1; i++)
        {
            arr1[i] = on1.nextInt();
        }

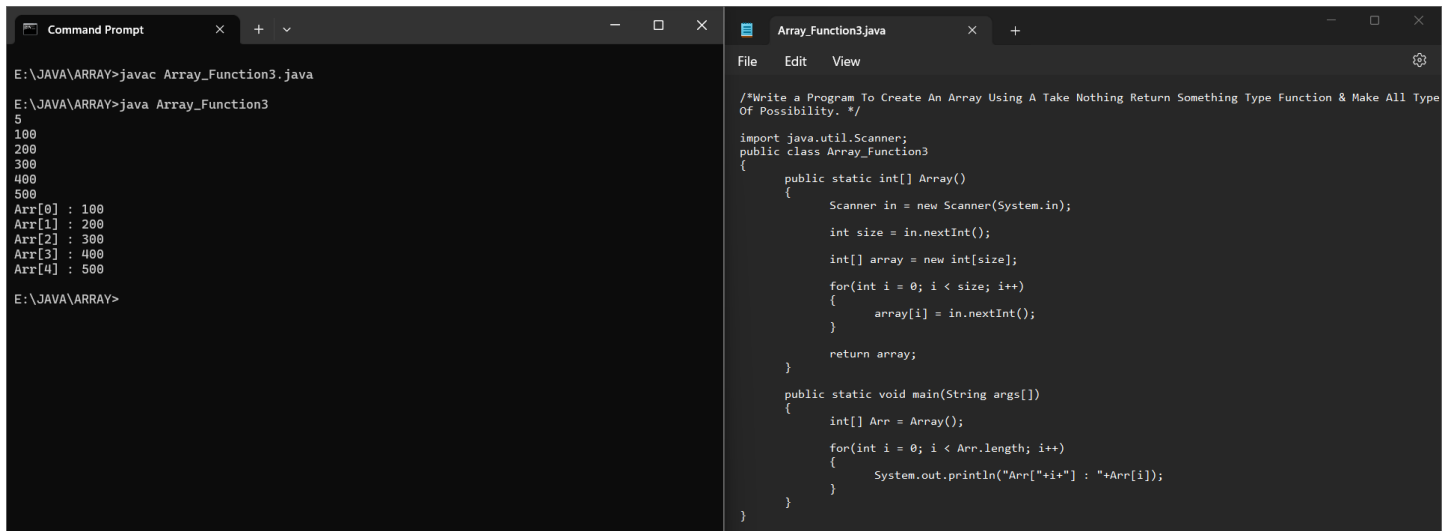
        for(int i = 0; i < sz1; i++)
        {
            System.out.println("Arr["+i+"] : "+arr1[i]);
        }
    }

    public static void main(String args[])
    {
        Scanner in = new Scanner(System.in);

        int size = in.nextInt();
        Array1(size);

        int size1 = in.nextInt();
        int[] ar1 = new int[size1];
        Array2(size1, ar1);
    }
}
```

Take Something Return Nothing Type Function & Array:



```
Command Prompt
E:\JAVA\ARRAY>javac Array_Function3.java
E:\JAVA\ARRAY>java Array_Function3
5
100
200
300
400
500
Arr[0] : 100
Arr[1] : 200
Arr[2] : 300
Arr[3] : 400
Arr[4] : 500
E:\JAVA\ARRAY>
```

```
Array_Function3.java
File Edit View
/*Write a Program To Create An Array Using A Take Nothing Return Something Type Function & Make All Type
Of Possibility. */
import java.util.Scanner;
public class Array_Function3
{
    public static int[] Array()
    {
        Scanner in = new Scanner(System.in);
        int size = in.nextInt();

        int[] array = new int[size];

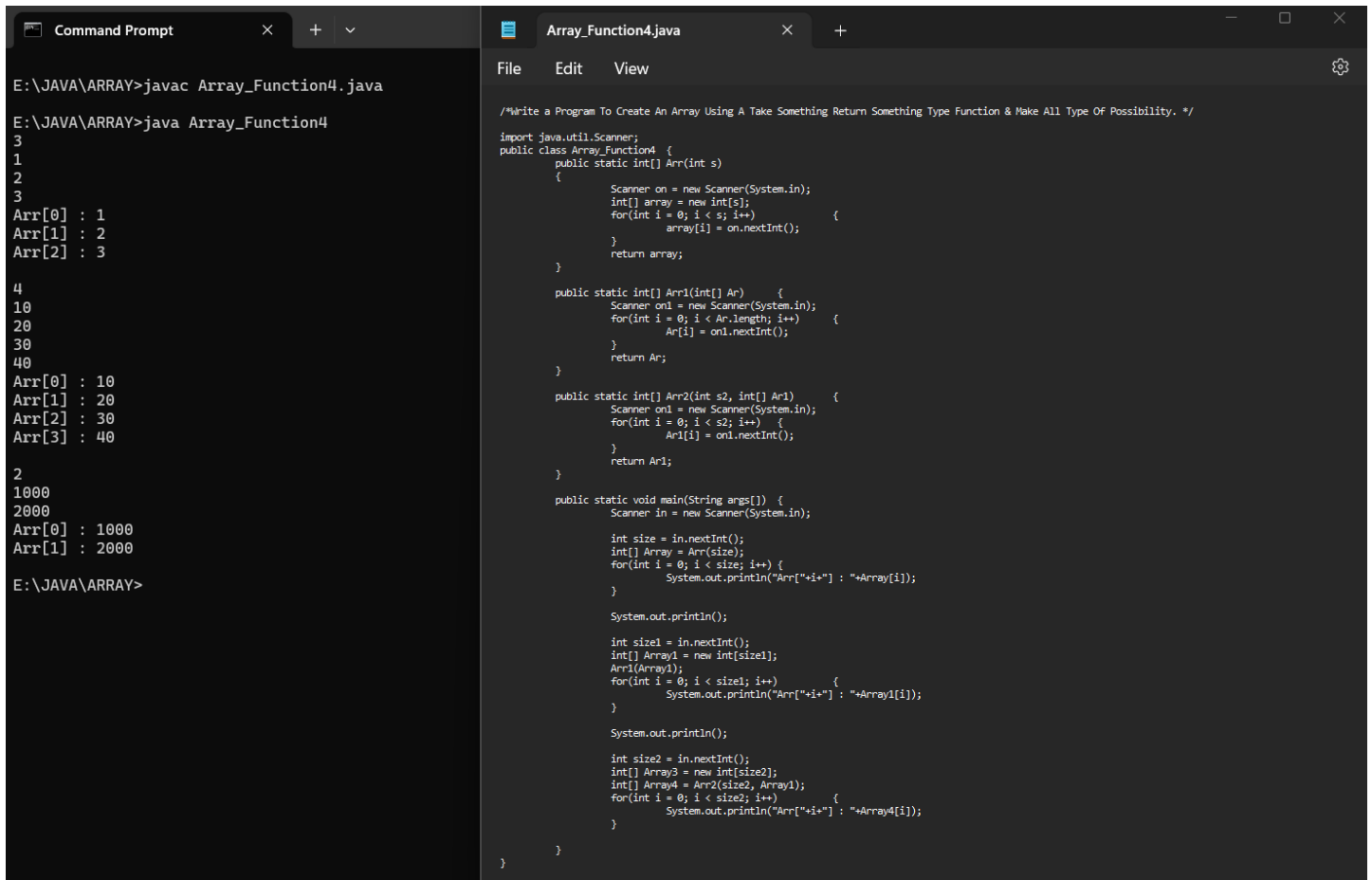
        for(int i = 0; i < size; i++)
        {
            array[i] = in.nextInt();
        }

        return array;
    }

    public static void main(String args[])
    {
        int[] Arr = Array();

        for(int i = 0; i < Arr.length; i++)
        {
            System.out.println("Arr["+i+"] : "+Arr[i]);
        }
    }
}
```

Take Something Return Something Type Function & Array:



```
Command Prompt
E:\JAVA\ARRAY>javac Array_Function4.java
E:\JAVA\ARRAY>java Array_Function4
3
1
2
3
Arr[0] : 1
Arr[1] : 2
Arr[2] : 3

4
10
20
30
40
Arr[0] : 10
Arr[1] : 20
Arr[2] : 30
Arr[3] : 40

2
1000
2000
Arr[0] : 1000
Arr[1] : 2000
E:\JAVA\ARRAY>
```

```
Array_Function4.java
File Edit View
/*Write a Program To Create An Array Using A Take Something Return Something Type Function & Make All Type Of Possibility. */
import java.util.Scanner;
public class Array_Function4 {
    public static int[] Arr(int s)
    {
        Scanner on = new Scanner(System.in);
        int[] array = new int[s];
        for(int i = 0; i < s; i++) {
            array[i] = on.nextInt();
        }
        return array;
    }

    public static int[] Arr1(int[] Ar) {
        Scanner on1 = new Scanner(System.in);
        for(int i = 0; i < Ar.length; i++) {
            Ar[i] = on1.nextInt();
        }
        return Ar;
    }

    public static int[] Arr2(int s2, int[] Ar1) {
        Scanner on1 = new Scanner(System.in);
        for(int i = 0; i < s2; i++) {
            Ar1[i] = on1.nextInt();
        }
        return Ar1;
    }

    public static void main(String args[]) {
        Scanner in = new Scanner(System.in);

        int size = in.nextInt();
        int[] Array = Arr(size);
        for(int i = 0; i < size; i++) {
            System.out.println("Arr["+i+"] : "+Array[i]);
        }

        System.out.println();

        int size1 = in.nextInt();
        int[] Array1 = new int[size1];
        Arr1(Array1);
        for(int i = 0; i < size1; i++) {
            System.out.println("Arr["+i+"] : "+Array1[i]);
        }

        System.out.println();

        int size2 = in.nextInt();
        int[] Array3 = new int[size2];
        int[] Array4 = Arr2(size2, Array1);
        for(int i = 0; i < size2; i++) {
            System.out.println("Arr["+i+"] : "+Array4[i]);
        }
    }
}
```

ARRAY METHODS IN JAVA:

```
Command Prompt
C:\Users\Atish kumar sahu\desktop>javac array.java
C:\Users\Atish kumar sahu\desktop>java array
Enter Range Value :
10
Enter Elements In Array :
51 33 48 12 28 44 11 79 19 35
Enter Element To Search :
44
Elements At : 5

array.java
import java.util.Scanner;
import java.util.Arrays;
public class array
{
    public static void main(String args[])
    {
        Scanner in = new Scanner(System.in);

        System.out.println("Enter Range Value : ");
        int size = in.nextInt();

        int[] arr = new int[size];

        System.out.println("Enter Elements In Array : ");
        for(int i = 0; i < size; i++)
        {
            arr[i] = in.nextInt();
        }

        System.out.println("Enter Element To Search :");
        int ele = in.nextInt();
        System.out.println("Elements At : "+Arrays.binarySearch(arr,ele));
    }
}
```

```
Command Prompt
C:\Users\Atish kumar sahu\desktop>javac array.java
C:\Users\Atish kumar sahu\desktop>java array
Enter Range Value :
6
Enter Elements In Array :
10 20 30 40 50 60
Copy Of Array :
Copy Array : [10, 20, 30, 40, 50, 60, 0, 0, 0, 0]

array.java
import java.util.Scanner;
import java.util.Arrays;
public class array
{
    public static void main(String args[])
    {
        Scanner in = new Scanner(System.in);

        System.out.println("Enter Range Value : ");
        int size = in.nextInt();

        int[] arr = new int[size];

        System.out.println("Enter Elements In Array : ");
        for(int i = 0; i < size; i++)
        {
            arr[i] = in.nextInt();
        }

        System.out.println("Copy Of Array :");
        System.out.println("Copy Array : "+Arrays.toString(Arrays.copyOf(arr,10)));
    }
}
```

```
Command Prompt
C:\Users\Atish kumar sahu\desktop>javac array.java
C:\Users\Atish kumar sahu\desktop>java array
Enter Range Value :
6
Enter Elements In Array :
18 7 11 9 3 20
Sort Of Array :
3 7 9 11 18 20

array.java
import java.util.Scanner;
import java.util.Arrays;
public class array
{
    public static void main(String args[])
    {
        Scanner in = new Scanner(System.in);

        System.out.println("Enter Range Value : ");
        int size = in.nextInt();

        int[] arr = new int[size];

        System.out.println("Enter Elements In Array : ");
        for(int i = 0; i < size; i++)
        {
            arr[i] = in.nextInt();
        }

        System.out.println("Sort Of Array :");
        Arrays.sort(arr);
        for(int i = 0; i < size; i++)
        {
            System.out.print(arr[i]+" ");
        }
    }
}
```

The End

Data Structure & Algorithm

The term means a value or set of values. It specifies either the value of a variable or a constant. While a data item that does not have subordinate data items is categorized and an elementary item, the one that is composed of one or more subordinate data items is called a group item.

Data is defined as collection of raw facts which do not have any defined context. We can say that Data is collection of numbers and text values which do not have any direct benefit to the organization that collects the data since it is unorganized and unprocessed . For example data collection done by government regarding health, income, dietary habits, family etc. does not have any benefit if it is stored as it is. The reason is that it is very voluminous and do not give any information. The Data thus collected needs to be processed before it can be used for decision making or reporting tasks.

A field is defined as a unit of meaningful information about an entity like date of flight, name of passenger, address etc. A record is a collection of data items. Record is a collection of units of information about a particular entity. Passenger of an airplane, an employee of an organization or an article sold from a store. A file is a collection of related records. A collection of records involving a set of entities with certain aspects in common and organized for some particular purpose is called a file. for example collection of records of all passengers.

A data structure is defined as mathematical or logical model used to organize and manipulate data. The management of data is done through various operations like traversal, insertion, deletion, searching, sorting etc. The logical organization of data as a data structure is done to make the data organization easier, allow access to individual elements of a data structure, defining the association among data elements and provide various operations to process the data to derive information.

To create a computer program and solve a specific problem, the programmer needs some structure to hold the data values. So, data structures are an important aspect of problem solving through computer programming. It must be easy to understand and implement and must exhibit the relationship among data elements required to provide solution. For example to implement a transportation or network problem a graph like data structure is needed. To

implement sequential execution of submitted tasks a queue like data structure is the best option.

Linear Data Structure:

Data elements are organized in a linear fashion in a linear data structure. Traversal can only be done sequentially. All the previous elements must be followed first to reach a particular element in linear data structures. It also means that only one data element can be directly reached from the previous element. Every data element in a linear data structure have a direct relation with its prior and following element. Examples of linear data structure are Arrays, Linked Lists, Stacks and Queues. Linear data structures can be represented in memory in two different ways. One way is to have to a linear relationship between elements by means of sequential memory locations. The other way is to have a linear relationship between elements by means of links.

Non-Linear Data Structure:

In a non-linear data structure data elements are arranged non-linearly and traversal cannot be done sequentially. Every data element may be linked to more than one data elements. The linkages of the data elements reflect a particular relationship. The relationship between the elements can be hierarchical or random. Examples of linear data structure are Trees and Graphs. if the elements of a data structure are not stored in a sequential order, then it is a non-linear data structure. The relationship of adjacency is not maintained between elements of a non-linear data structure.

Primitive Data Structure:

Primitive data structures are the fundamental data types which are supported by a programming language. Some basic data types are integer, real, character, and boolean. The terms 'data type', 'basic data type', and 'primitive data type' are often used interchangeably.

Non-Primitive Data Structure:

Non-primitive data structures are those data structures which are created using primitive data structures. Examples of such data structures include linked lists, stacks, trees, and graphs. Non-primitive data structures can further be classified into two categories: linear and non-linear data structures.

Linear Data Structure Elements:

Array:

An array is an indexed collection of fixed number of homogeneous data elements. The main advantage of arrays is we can represent huge number of values by using single variable. So that readability of the code will be improved. But the main disadvantage of array is fixed in size that is one's we create an array there is no chance of increasing or decreasing in the size based on our requirement.

Hence to use arrays concept compulsory we should know the size in advance, which may not possible always. An array is a homogeneous and linear data structure that is stored in contiguous memory locations. An element in an array can be accessed, inserted or removed by specifying its position or index or subscript along with the name of the array.

Linked List:

A linked list is a fundamental data structure in computer science, used to organize and manage collections of elements. Unlike arrays, which store elements in contiguous memory locations, a linked list consists of nodes, each containing both data and a reference (or pointer) to the next node in the sequence. This flexible structure enables dynamic memory allocation, making it efficient for inserting and deleting elements anywhere in the list. However, traversal through a linked list can be slower compared to arrays, as it requires following the links sequentially.

Linked lists come in various forms, including singly linked lists with one-way connections, doubly linked lists with both next and previous references, and circular linked lists where the last node points back to the first. Each variation offers specific advantages depending on the use case, making linked lists a versatile and essential component of data structuring and manipulation in programming. These are the following types of Linked List.

Singly Linked List:

Singly Linked List is a type of linked list when each node contains data and a reference to the next node. It's a simple structure that allows traversal in only one direction, from the head (start) to the tail (end) of the list. Singly linked lists are efficient for insertions and deletions at the beginning and middle of the list, but accessing elements further down the list requires linear traversal.

A linear or singly linked list is a linear data structure consisting of a sequence of nodes. Each node consists of two parts – Data and Link to the next node. The address of first node is stored in a pointer Start.

Each node stores the address of its next node in its link part. The linked list is non-contiguous collection of nodes that allows insertion and deletion at any specific location.

Doubly Linked List:

A doubly linked list is a linear data structure consisting of a sequence of nodes. Each node consists of three parts – Data, Forward Pointer to the next node and Backward Pointer to the previous node. The address of first node is stored in a pointer called the Header and address of last node is stored in pointer called the Trailer. The doubly linked list allows traversal in forward and backward direction.

Doubly Linked List is a type of linked list where each node has references to both the next and the previous nodes. This bidirectional linkage allows for easier traversal in both directions, making insertions and deletions at any position more efficient. However, doubly linked lists require more memory due to the additional references in each node.

Circular Linked List:

A circular linked list forms a closed loop, where the last node points back to the first node, creating a circular structure. This type can be implemented as either singly or doubly linked. Circular linked lists are useful for applications like implementing circular buffers or managing processes in a round-robin scheduling algorithm.

Queue:

A queue is a linear data structure that follows the First-In-First-Out (FIFO) principle. Elements are added to the back (enqueue) and removed from the front (dequeue). It resembles a real-world queue, like people waiting in line. Queues are useful for managing tasks in order, such as in breadth-first searches and print job scheduling.

Queue is a linear data structure that follows the first-in first-out scheme. Insertions are done at the rear of the queue and removals at the front of the queue. Two variables needed for these operations are the front and rear.

Circular Queue:

A Circular Queue is queue following “First In First Out” (FIFO) principle with the difference that the last element is associated back with the first element of the queue.

Insertion will be done at end called the REAR. Deletion will take place at beginning called the REAR. The elements that are inserted earlier will be deleted first. The elements added later will be deleted later. An item is added at the end and removed from the beginning and the first follows the last element. It is used to maintain sequence of events in a round robin fashion.

Double-Ended Queue:

A variation of Queue is Dequeue in which deletions and insertions can be done at both ends. A deque is a versatile linear structure that supports insertion and deletion at both ends. It can function as a queue or a stack, providing flexibility for various applications. Deques are efficient for tasks like implementing algorithms that require simultaneous access to both ends, such as palindrome checking and implementing sliding window problems.

Priority Queue:

A priority queue is a data structure where elements have assigned priorities, and the highest-priority element is accessed first. It doesn't follow a strict order like queues. It's often implemented as a binary heap, enabling efficient insertion and retrieval of the highest-priority element. Priority queues are crucial in scenarios where tasks must be processed based on urgency or importance, such as in Dijkstra's algorithm for shortest path finding and Huffman coding in data compression.

priority queue is a linear data structure that behaves like a queue where each element has a priority associated with it on the basis of which deletions are done. The element with maximum (minimum) priority will be deleted from the queue. It can also be defined as a data structure that supports operations of min or max insert, delete or search.

Non-Linear Data Structure Element:

Heap:

A max(min) heap is a complete binary tree with the property that the value of each node is at least as large(small) as the values at its children. If the elements are distinct then the root node will be the largest (smallest).

The insertion will always be done as a leaf node first and then the heap will be “happified” to bring the newly inserted node at its correct position. Heapify means that the node will be compared with its parent and if it is larger(smaller) than its parent then there will be an interchange between parent node and the newly inserted child node. This process will end up in a heap after insertion.

Deletion will be done at the root node. The last node of the complete binary tree will replace the deleted root node and then the process of Heapify will be done starting from the new root node.

Tree:

A tree is a Non-Linear Data Structure that is an abstract model of a hierarchical structure consisting of nodes with a parent-child relation. Its applications are Organization charts, File systems, Programming environments. There are four things associated with any tree -Distinction between nodes, Value of nodes, orientation structure and the number of levels.

Root is starting point of the tree is a node called root characterized by a node without parent. External/Leaf Node is a node without any children. Internal node is all the nodes other than the root leaf nodes. It can be said to be a node with at least one child. Ancestors of a node parent, grand-parent and any other nodes which lie on the path from root to that node.

Depth of a node is the number of ancestors for any give node. Height of a tree is maximum depth among all the leaf nodes. Descendant of a node is child grandchild and all the other nodes lying on path from a node to a leaf node. Subtree is a tree consisting of a node and its descendants.

Graph:

A graph $G(V,E)$ is a defined with two sets. V , a set of vertices, and E the set of edges between two pair of vertices from the set V . When the edges of the graph have to be defined with the direction, it is called a directed graph or digraph, and the edges are called directed edges or arcs. In a digraph edge

(a,b is not the same as edge (b, a). In a directed edge (a,b) a is called the source/starting point and b is called the sink/destination/ end point. In an undirected graph the direction of edge is not specified.

When the edges of the graph have to be defined with the direction, it is called a directed graph or digraph, and the edges are called directed edges or arcs. In a digraph edge (a,b is not the same as edge (b, a). In a directed edge (a,b) a is called the source/starting point and b is called the sink/destination/ end point. In an undirected graph the direction of edge is not specified. Two vertices that are connected to each other with an edge are called neighbors. They are also said to be adjacent to each other.

Data Structure Operations:

Traversal:

Traversal of a data structure can be defined as the process of visiting every element of a data structure at least once. This operation is most commonly used for printing, searching, displaying or reading the elements stored in the data structure. It is usually done by using a variable or pointer that indicates the current element of the data structure being processed.

This variable or pointer is updated after visiting an element so that the location or address of next element can be found. When the last element of the data structure is reached the traversal process ends. Traversal can be useful when all the elements of the data structure have to be manipulated in similar way.

In an array traversal begins from the first element. A variable stores the index of first element of the array and it is incremented after visiting each node. When the value of this variable is equal to the index of last node, the traversal of the array ends.

Insertion:

Once a data structure is created, it can be extended by adding new elements. The process of adding new elements in an existing data structure is called insertion. Where the element can be added depends upon the data structure that a user is dealing with. Insertion operation always ends up in increasing the size of the data structure.

A linked list and an array allow a user to insert a new element at any location. A stack and a queue allow a user to insert a new element only at a specific end. New nodes can be added to a graph or tree in a random fashion.

For all the data structure the insertion can be done till the data structure has enough space to store new elements either due to its defined size or memory availability. A condition when a user tries to insert a new element in a data structure that does not have the needed space for new element is called Overflow

Deletion:

Once a data structure is created, a user can remove any of the existing elements and free up the space occupied by it. The process of deleting an existing element from a data structure is called deletion. Which element can be deleted depends upon the data structure that a user is dealing with. Deletion operation always ends up in reducing the size of the data structure.

A linked list and an array allow a user to delete an existing element at any location i.e. start, mid or end. A stack and a queue allow a user to delete an element only at a specific end. Nodes can be deleted from a graph or tree in a random fashion.

For all the data structure the deletion can be done till the data structure has elements. A condition when a user tries to delete an element from a data structure that does not have any element is called Underflow

Search:

The process of locating an element in data structure and returning its index or address is called Search. This is the most commonly used operation in a data structure. Since a data structure stores data in an organized fashion for convenient processing, it is important that an element could be easily located in a data structure.

When performing search in a data structure a key value is needed, this is matched with the values stored in the data structure. When the value of the element matches the key value successful search is done and the search operation returns the location of that element. If the key value does not match any element in the data structure and reaches end, it is unsuccessful search and a null location is returned as a result of search operation.

Sorting:

The process of arranging the data elements in a data structure in a specific order (ascending or descending) by specific key values is called sorting. The students records stored in an array can be sorted by their registration

numbers, names or the scores. In each sorting the criteria will be different to fulfill the processing needs of a user.

Sorting may result in physical relocation of elements or it can be just a rearrangement of key values as index without changing the physical address of complete data element so that a subsequent traversal operation displays the data in sorted manner.

Merge:

Merging can be defined as the process of combining elements of two data structures. In a simpler form merging can be treated as appending set of elements of one data structure after elements of another data structure of same element structure. For example two arrays containing student data of two different classes with same fields to form one list. The final data structure may or may not be sorted.

Another form of merging can be to merge two data structure of different constructions to create totally new data structure. For example one data structure with student registration number and name can be merged with another data structure containing course and result data of same set of students to form one list(or file)

Copy:

The process of copying is the operation that makes a new data structure from an existing data structure. In the process of copying the original data structure retains its structure and data elements. The new data structure has same data elements. Both these data structure can be used to perform all the operations discussed previously. Changes in one data structure will not affect the elements or count of elements of the other data structure.

Array Methods:

Array length method:

The above method is used for to find the length of an array.

<pre>C:\Users\Atish kumar sahu\desktop>javac test.java C:\Users\Atish kumar sahu\desktop>java test Enter Size Value : 10 Enter Elements : 10 20 30 40 50 60 70 80 90 100 Array Is : 10 20 30 40 50 60 70 80 90 100 length : 10 C:\Users\Atish kumar sahu\desktop></pre>	<pre>File Edit View import java.util.*; public class test { public static void main(String args[]) { Scanner in = new Scanner(System.in); System.out.println("Enter Size Value : "); int size = in.nextInt(); int[] ar = new int[size]; System.out.println("Enter Elements : "); for(int i = 0; i < size; i++) { ar[i] = in.nextInt(); } System.out.println("Array Is : "); for(int i = 0; i < size; i++) { System.out.print(ar[i]+" "); } System.out.println("\nlength : "+ar.length); } }</pre>
--	---

Array clone method:

The above method clone or copy the array.

<pre>C:\Users\Atish kumar sahu\desktop>javac test.java C:\Users\Atish kumar sahu\desktop>java test Enter Size Value : 9 Enter Elements : 10 20 30 40 50 60 70 80 90 Array Is : 10 20 30 40 50 60 70 80 90 clone array : 10 20 30 40 50 60 70 80 90 C:\Users\Atish kumar sahu\desktop></pre>	<pre>File Edit View import java.util.*; public class test { public static void main(String args[]) { Scanner in = new Scanner(System.in); System.out.println("Enter Size Value : "); int size = in.nextInt(); int[] ar = new int[size]; System.out.println("Enter Elements : "); for(int i = 0; i < size; i++) { ar[i] = in.nextInt(); } System.out.println("Array Is : "); for(int i = 0; i < size; i++) { System.out.print(ar[i]+" "); } int[] clonea = new int[size]; clonea = ar.clone(); System.out.println("\nclone array : "); for(int i = 0; i < size; i++) { System.out.print(clonea[i]+" "); } } }</pre>
---	--

equals() method:

The above method is used for to check whether two arrays are equals or not.

<pre>C:\Users\Atish kumar sahu\desktop>javac test.java C:\Users\Atish kumar sahu\desktop>java test Enter Size Value : 5 Enter Elements : 10 20 30 40 50 Array Is : 10 20 30 40 50 ar equals ar1 : true ar equals ar2 : false ar1 equals ar2 : false C:\Users\Atish kumar sahu\desktop></pre>	<pre>File Edit View import java.util.*; public class test { public static void main(String args[]) { Scanner in = new Scanner(System.in); System.out.println("Enter Size Value : "); int size = in.nextInt(); int[] ar = new int[size]; System.out.println("Enter Elements : "); for(int i = 0; i < size; i++) { ar[i] = in.nextInt(); } System.out.println("Array Is : "); for(int i = 0; i < size; i++) { System.out.print(ar[i]+" "); } int[] ar1 = {10, 20, 30, 40, 50}; int[] ar2 = {60, 70, 80, 90, 100}; System.out.println("\nar equals ar1 : "+Arrays.equals(ar,ar1)); System.out.println("ar equals ar2 : "+Arrays.equals(ar,ar2)); System.out.println("ar1 equals ar2 : "+Arrays.equals(ar1,ar2)); } }</pre>
---	---

copyOf() methods:

The Java Arrays copyOf(int[] original,int newLength) method copies the specified array, truncating or padding with false (if necessary) so the copy has the specified length.

<pre>C:\Users\Atish kumar sahu\desktop>javac test.java C:\Users\Atish kumar sahu\desktop>java test Enter Size Value : 7 Enter Elements : 10 20 30 40 50 60 70 Array Is : 10 20 30 40 50 60 70 copy array : [10, 20, 30, 40, 50, 60, 70, 0, 0, 0] C:\Users\Atish kumar sahu\desktop></pre>	<pre>File Edit View import java.util.*; public class test { public static void main(String args[]) { Scanner in = new Scanner(System.in); System.out.println("Enter Size Value : "); int size = in.nextInt(); int[] ar = new int[size]; System.out.println("Enter Elements : "); for(int i = 0; i < size; i++) { ar[i] = in.nextInt(); } System.out.println("Array Is : "); for(int i = 0; i < size; i++) { System.out.print(ar[i]+" "); } System.out.println("\ncopy array : "+Arrays.toString(Arrays.copyOf(ar, 10))); } }</pre>
--	--

HashCode: Returns an integer hashCode of this array instance.

<pre>C:\Users\Atish kumar sahu\desktop>javac test.java C:\Users\Atish kumar sahu\desktop>java test Enter Size Value : 8 Enter Elements : 10 20 30 40 50 60 70 80 Array Is : 10 20 30 40 50 60 70 80 HashCode : -91651031 C:\Users\Atish kumar sahu\desktop></pre>	<pre>File Edit View import java.util.*; public class test { public static void main(String args[]) { Scanner in = new Scanner(System.in); System.out.println("Enter Size Value : "); int size = in.nextInt(); int[] ar = new int[size]; System.out.println("Enter Elements : "); for(int i = 0; i < size; i++) { ar[i] = in.nextInt(); } System.out.println("Array Is : "); for(int i = 0; i < size; i++) { System.out.print(ar[i]+" "); } System.out.println("\nHashCode : "+Arrays.hashCode(ar)); } }</pre>
--	---

identityHashCode(): To find the address of each element of an array elements.

<pre>C:\Users\Atish kumar sahu\desktop>javac test.java C:\Users\Atish kumar sahu\desktop>java test Enter Size Value : 6 Enter Elements : 10 20 30 40 50 60 Array Is : ar[0] : 10 hashCode : 1118140819 ar[1] : 20 hashCode : 1975012498 ar[2] : 30 hashCode : 1808253012 ar[3] : 40 hashCode : 589431969 ar[4] : 50 hashCode : 1252169911 ar[5] : 60 hashCode : 2101973421 C:\Users\Atish kumar sahu\desktop></pre>	<pre>File Edit View import java.util.*; public class test { public static void main(String args[]) { Scanner in = new Scanner(System.in); System.out.println("Enter Size Value : "); int size = in.nextInt(); int[] ar = new int[size]; System.out.println("Enter Elements : "); for(int i = 0; i < size; i++) { ar[i] = in.nextInt(); } System.out.println("Array Is : "); for(int i = 0; i < size; i++) { int address = System.identityHashCode(ar[i]); System.out.println("ar["+i+"] : "+ar[i]+" hashCode : "+address); } } }</pre>
--	--

sort() method: to sort an array in ascending order.

<pre>C:\Users\Atish kumar sahu\desktop>javac test.java C:\Users\Atish kumar sahu\desktop>java test Enter Size Value : 5 Enter Elements : 90 80 70 60 50 UnSorted Array Is : [90, 80, 70, 60, 50] Sorted Array Is : [50, 60, 70, 80, 90] C:\Users\Atish kumar sahu\desktop></pre>	<pre>File Edit View import java.util.*; public class test { public static void main(String args[]) { Scanner in = new Scanner(System.in); System.out.println("Enter Size Value : "); int size = in.nextInt(); int[] ar = new int[size]; System.out.println("Enter Elements : "); for(int i = 0; i < size; i++) { ar[i] = in.nextInt(); } System.out.println("UnSorted Array Is : "+Arrays.toString(ar)); Arrays.sort(ar); System.out.println("Sorted Array Is : "+Arrays.toString(ar)); } }</pre>
---	--

<pre>C:\Users\Atish kumar sahu\desktop>javac test.java C:\Users\Atish kumar sahu\desktop>java test Enter Size Value : 7 Enter Element : A a B b C c D Unsorted Array : [A, a, B, b, C, c, D] Sorted Array : [A, B, C, D, a, b, c] C:\Users\Atish kumar sahu\desktop>java test Enter Size Value : 8 Enter Element : Atish Lipun Katayani Soni Sahu Verma Billu Billa Unsorted Array : [Atish, Lipun, Katayani, Soni, Sahu, Verma, Billu, Billa] Sorted Array : [Atish, Billa, Billu, Katayani, Lipun, Sahu, Soni, Verma] C:\Users\Atish kumar sahu\desktop></pre>	<pre>File Edit View import java.util.*; public class test { public static void main(String args[]) { Scanner in = new Scanner(System.in); System.out.println("Enter Size Value : "); int size = in.nextInt(); String[] ar = new String[size]; System.out.println("Enter Element : "); for(int i = 0; i < size; i++) { ar[i] = in.next(); } System.out.println("Unsorted Array : "+Arrays.toString(ar)); Arrays.sort(ar); System.out.println("Sorted Array : "+Arrays.toString(ar)); } }</pre>
---	--

sort() method: Descending Order

<pre>test.java File Edit View import java.util.*; class MyComparator implements Comparator{ public int compare(Object o1, Object o2) { String s1 = o1.toString(); String s2 = o2.toString(); return s2.compareTo(s1); } } public class test { public static void main(String args[]) { String[] s = {"A", "B", "a", "C", "D", "d", "b", "c"}; System.out.println("String Array Is : "); for(String s1 : s){ System.out.print(s1+" "); } Arrays.sort(s, new MyComparator()); System.out.println("\nDescending Sort : "); for(String s1 : s){ System.out.print(s1+" "); } } }</pre>	<pre>Command Prompt C:\Users\Atish kumar sahu\desktop>javac test.java Note: test.java uses unchecked or unsafe operations. Note: Recompile with -Xlint:unchecked for details. C:\Users\Atish kumar sahu\desktop>java test String Array Is : A B a C D d b c Descending Sort : d c b a D C B A C:\Users\Atish kumar sahu\desktop></pre>
---	---

```
test.java
File Edit View
import java.util.*;
public class test {
    public static void main(String args[]) {
        int[] a = {40,10,50,70,80,20,30,60};
        System.out.println("Array Is :");
        for(int a1 : a){
            System.out.print(a1+" ");
        }
        Arrays.sort(a);
        System.out.println("\nDescending Order : ");
        for(int i = a.length - 1; i >= 0; i--) {
            System.out.print(a[i]+" ");
        }
    }
}

Command Prompt
C:\Users\Atish kumar sahu\desktop>javac test.java

C:\Users\Atish kumar sahu\desktop>java test
Array Is :
40 10 50 70 80 20 30 60
Descending Order :
80 70 60 50 40 30 20 10
C:\Users\Atish kumar sahu\desktop>
```

binarySearch(): Search the element from array.

```
C:\Users\Atish kumar sahu\desktop>javac test.java

C:\Users\Atish kumar sahu\desktop>java test
Enter Size Value :
7
Enter Elements :
11 12 13 14 15 16 17
Array : [11, 12, 13, 14, 15, 16, 17]
Search 12 : 1
Search 20 : -8
C:\Users\Atish kumar sahu\desktop>

File Edit View
import java.util.*;
public class test {
    public static void main(String args[]) {
        Scanner in = new Scanner(System.in);
        System.out.println("Enter Size Value : ");
        int size = in.nextInt();

        int[] ar = new int[size];
        System.out.println("Enter Elements : ");
        for(int i = 0; i < size; i++) {
            ar[i] = in.nextInt();
        }
        Arrays.sort(ar);
        System.out.println("Array : "+Arrays.toString(ar));
        System.out.println("Search 12 : "+Arrays.binarySearch(ar, 12));
        System.out.println("Search 20 : "+Arrays.binarySearch(ar, 20));
    }
}
```

```
C:\Users\Atish kumar sahu\desktop>javac test.java

C:\Users\Atish kumar sahu\desktop>java test
Enter Size Value :
7
Enter Elements :
d C b A e f G
Search A : 0
Search B : -2
C:\Users\Atish kumar sahu\desktop>

File Edit View
import java.util.*;
public class test {
    public static void main(String args[]) {
        Scanner in = new Scanner(System.in);
        System.out.println("Enter Size Value : ");
        int size = in.nextInt();

        String[] str = new String[size];
        System.out.println("Enter Elements : ");
        for(int i = 0; i < size; i++) {
            str[i] = in.next();
        }
        Arrays.sort(str);
        System.out.println("Search A : "+Arrays.binarySearch(str, "A"));
        System.out.println("Search B : "+Arrays.binarySearch(str, "B"));
    }
}
```

```
C:\Users\Atish kumar sahu\desktop>javac test.java
Note: test.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.

C:\Users\Atish kumar sahu\desktop>java test
Enter Size Value :
6
Enter Elements :
a b D A e C
Search D : 3
Search B : -6
C:\Users\Atish kumar sahu\desktop>

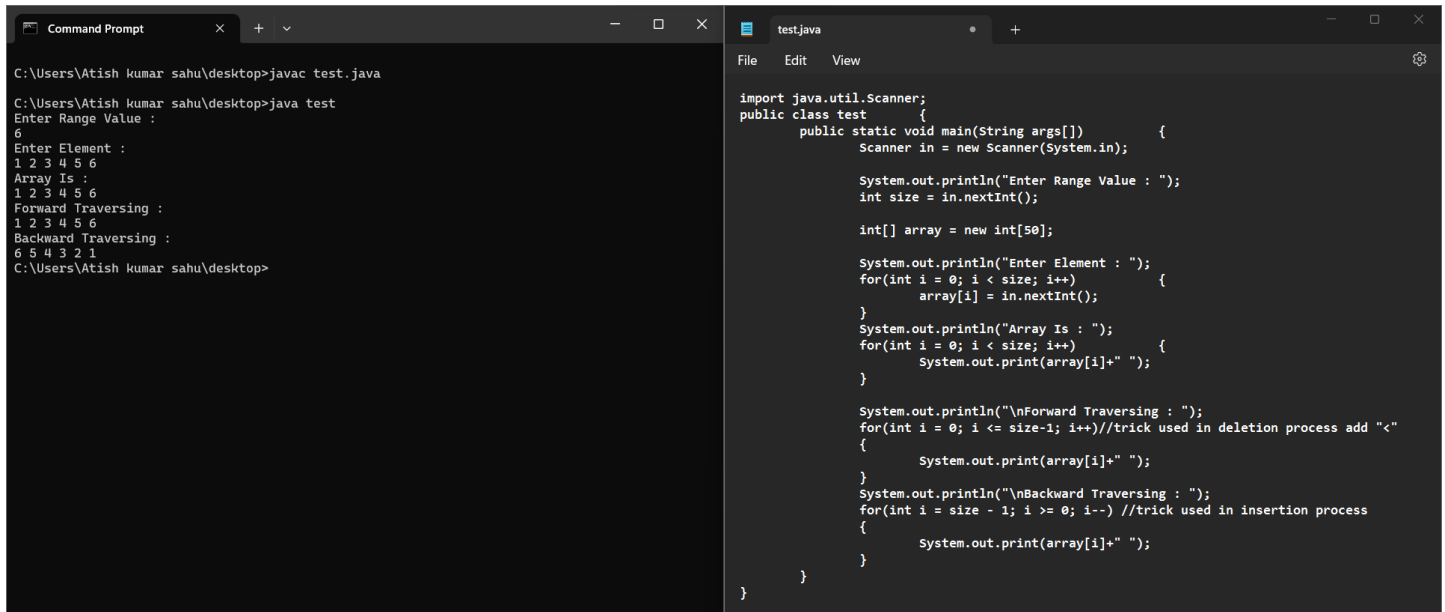
File Edit View
import java.util.*;
class MyComparator implements Comparator
{
    public int compare(Object o1, Object o2) {
        String s1 = o1.toString();
        String s2 = o2.toString();
        return s2.compareTo(s1);
    }
}
public class test {
    public static void main(String args[]) {
        Scanner in = new Scanner(System.in);
        System.out.println("Enter Size Value : ");
        int size = in.nextInt();

        String[] str = new String[size];
        System.out.println("Enter Elements : ");
        for(int i = 0; i < size; i++) {
            str[i] = in.next();
        }
        Arrays.sort(str, new MyComparator());
        System.out.println("Search D : "+Arrays.binarySearch(str, "D", new MyComparator()));
        System.out.println("Search B : "+Arrays.binarySearch(str, "B", new MyComparator()));
    }
}
```

Question1: Traversing Of Array

Take an array from user and traverse the array in forward and backward direction.

Input: [1 2 3 4 5 6] output: arr[1,2,3,4,5,6] arr[6,5,4,3,2,1]



The screenshot shows a Java program named `test.java` that takes an array from the user and traverses it in both forward and backward directions. The code is as follows:

```
import java.util.Scanner;
public class test
{
    public static void main(String args[])
    {
        Scanner in = new Scanner(System.in);

        System.out.println("Enter Range Value : ");
        int size = in.nextInt();

        int[] array = new int[50];

        System.out.println("Enter Element : ");
        for(int i = 0; i < size; i++)
        {
            array[i] = in.nextInt();
        }
        System.out.println("Array Is : ");
        for(int i = 0; i < size; i++)
        {
            System.out.print(array[i]+" ");
        }

        System.out.println("\nForward Traversing : ");
        for(int i = 0; i <= size-1; i++)//trick used in deletion process add "<"
        {
            System.out.print(array[i]+" ");
        }
        System.out.println("\nBackward Traversing : ");
        for(int i = size - 1; i >= 0; i--) //trick used in insertion process
        {
            System.out.print(array[i]+" ");
        }
    }
}
```

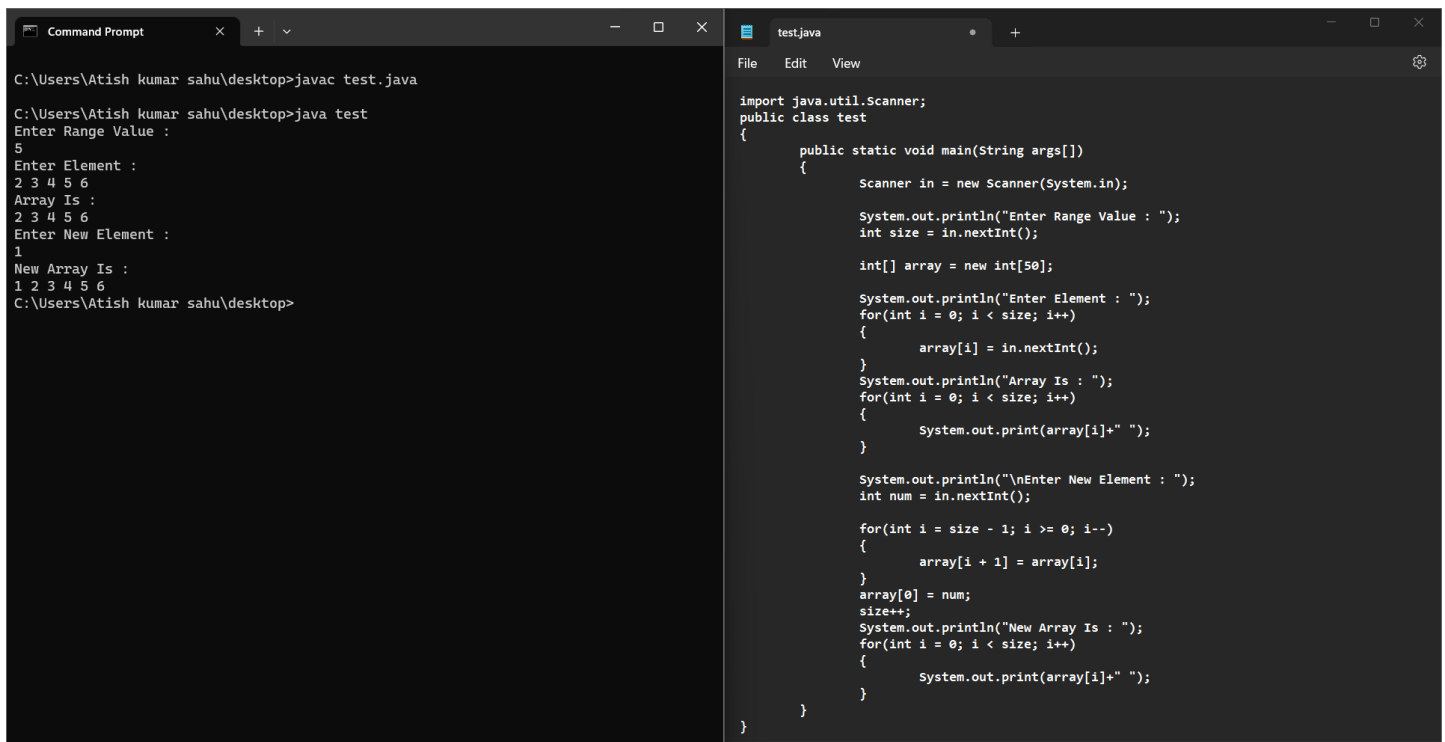
The execution output in the Command Prompt is as follows:

```
C:\Users\Atish kumar sahu\desktop>javac test.java
C:\Users\Atish kumar sahu\desktop>java test
Enter Range Value :
6
Enter Element :
1 2 3 4 5 6
Array Is :
1 2 3 4 5 6
Forward Traversing :
1 2 3 4 5 6
Backward Traversing :
6 5 4 3 2 1
C:\Users\Atish kumar sahu\desktop>
```

Question2: Insertion Of Element At First Index

Take an array from user and take a random value from the user and insert that value in the first index of the given array.

Input: arr [2 3 4 5 6] Num = 1 Output: arr[1,2,3,4,5,6]



The screenshot shows a Java program named `test.java` that inserts a new element at the first index of an array. The code is as follows:

```
import java.util.Scanner;
public class test
{
    public static void main(String args[])
    {
        Scanner in = new Scanner(System.in);

        System.out.println("Enter Range Value : ");
        int size = in.nextInt();

        int[] array = new int[50];

        System.out.println("Enter Element : ");
        for(int i = 0; i < size; i++)
        {
            array[i] = in.nextInt();
        }
        System.out.println("Array Is : ");
        for(int i = 0; i < size; i++)
        {
            System.out.print(array[i]+" ");
        }

        System.out.println("\nEnter New Element : ");
        int num = in.nextInt();

        for(int i = size - 1; i >= 0; i--)
        {
            array[i + 1] = array[i];
        }
        array[0] = num;
        size++;
        System.out.println("New Array Is : ");
        for(int i = 0; i < size; i++)
        {
            System.out.print(array[i]+" ");
        }
    }
}
```

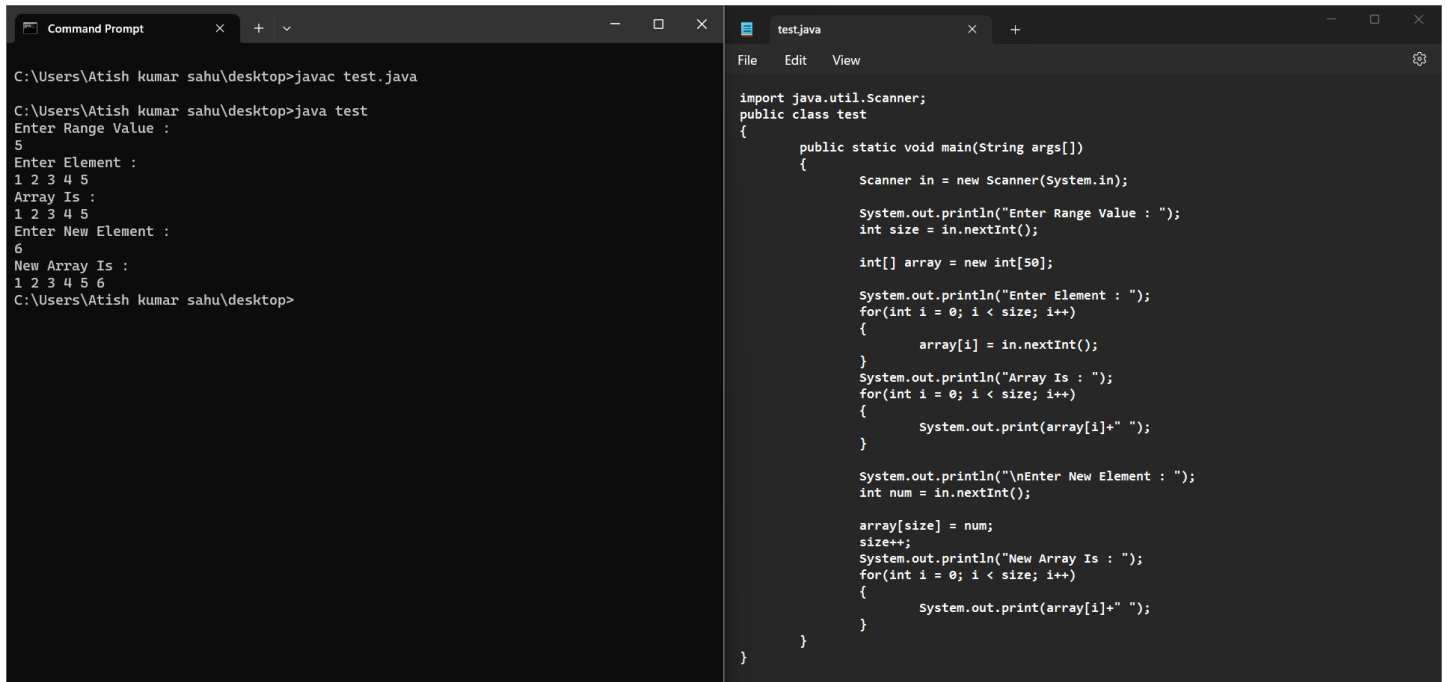
The execution output in the Command Prompt is as follows:

```
C:\Users\Atish kumar sahu\desktop>javac test.java
C:\Users\Atish kumar sahu\desktop>java test
Enter Range Value :
5
Enter Element :
2 3 4 5 6
Array Is :
2 3 4 5 6
Enter New Element :
1
New Array Is :
1 2 3 4 5 6
C:\Users\Atish kumar sahu\desktop>
```

Question3: Insertion Of Element In Last Index:

Take an array from user and take a random value from the user and insert that value in the last index of the given array.

Input: arr [1 2 3 4 5] Num = 6 Output: arr[1,2,3,4,5,6]



The screenshot shows two windows. The left window is a Command Prompt showing the execution of a Java program. The right window shows the source code of the program, `test.java`.

```
Command Prompt
C:\Users\Atish kumar sahu\desktop>javac test.java
C:\Users\Atish kumar sahu\desktop>java test
Enter Range Value :
5
Enter Element :
1 2 3 4 5
Array Is :
1 2 3 4 5
Enter New Element :
6
New Array Is :
1 2 3 4 5 6
C:\Users\Atish kumar sahu\desktop>
```

```
test.java
import java.util.Scanner;
public class test
{
    public static void main(String args[])
    {
        Scanner in = new Scanner(System.in);

        System.out.println("Enter Range Value : ");
        int size = in.nextInt();

        int[] array = new int[50];

        System.out.println("Enter Element : ");
        for(int i = 0; i < size; i++)
        {
            array[i] = in.nextInt();
        }
        System.out.println("Array Is : ");
        for(int i = 0; i < size; i++)
        {
            System.out.print(array[i]+" ");
        }

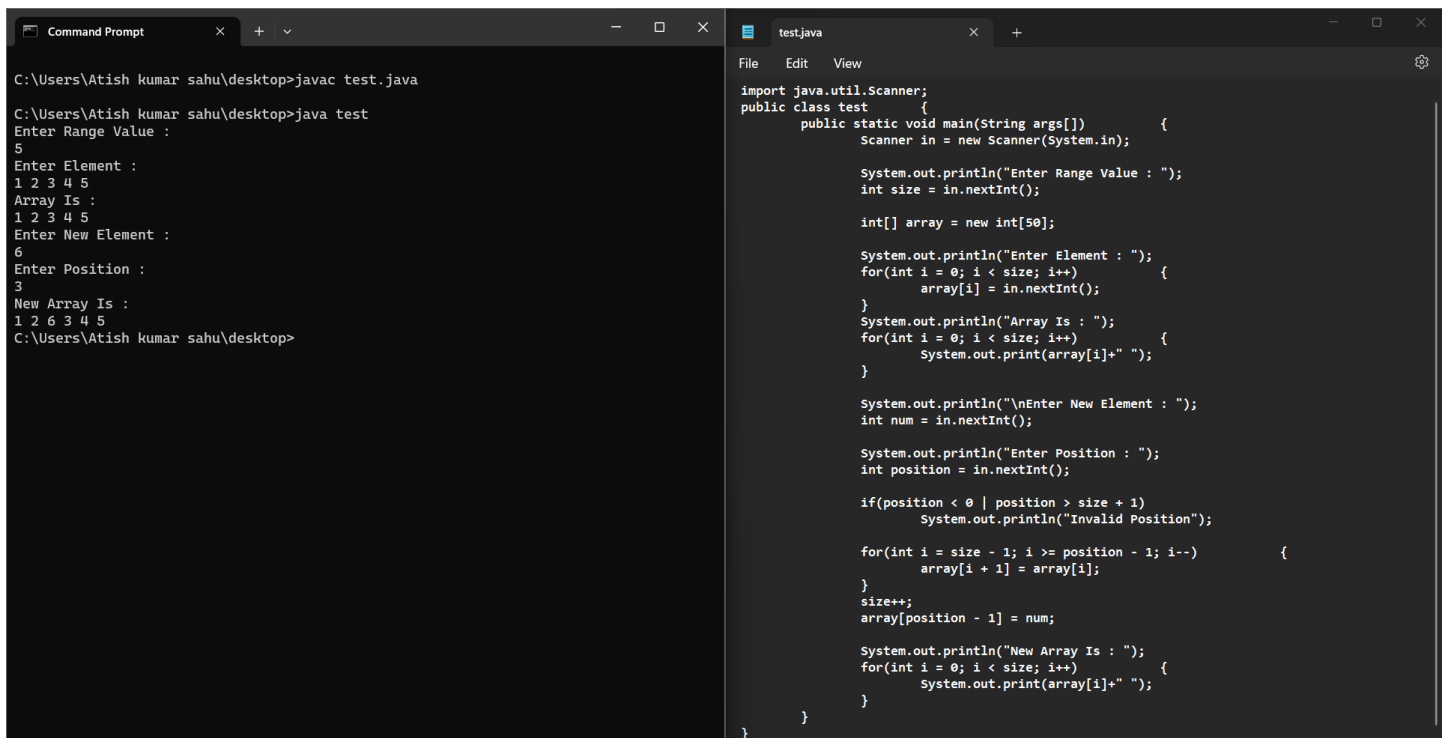
        System.out.println("\nEnter New Element : ");
        int num = in.nextInt();

        array[size] = num;
        size++;
        System.out.println("New Array Is : ");
        for(int i = 0; i < size; i++)
        {
            System.out.print(array[i]+" ");
        }
    }
}
```

Question4: Insertion Of Element In Specific Index:

Take an array from user and take a random value from the user and insert that value in the specific given index of the given array.

Input: arr [1 2 3 4 5] Num = 6 Index = 3 Output: arr[1,2,6,3,4,5]



The screenshot shows two windows. The left window is a Command Prompt showing the execution of a Java program. The right window shows the source code of the program, `test.java`.

```
Command Prompt
C:\Users\Atish kumar sahu\desktop>javac test.java
C:\Users\Atish kumar sahu\desktop>java test
Enter Range Value :
5
Enter Element :
1 2 3 4 5
Array Is :
1 2 3 4 5
Enter New Element :
6
Enter Position :
3
New Array Is :
1 2 6 3 4 5
C:\Users\Atish kumar sahu\desktop>
```

```
test.java
import java.util.Scanner;
public class test
{
    public static void main(String args[])
    {
        Scanner in = new Scanner(System.in);

        System.out.println("Enter Range Value : ");
        int size = in.nextInt();

        int[] array = new int[50];

        System.out.println("Enter Element : ");
        for(int i = 0; i < size; i++)
        {
            array[i] = in.nextInt();
        }
        System.out.println("Array Is : ");
        for(int i = 0; i < size; i++)
        {
            System.out.print(array[i]+" ");
        }

        System.out.println("\nEnter New Element : ");
        int num = in.nextInt();

        System.out.println("Enter Position : ");
        int position = in.nextInt();

        if(position < 0 || position > size + 1)
            System.out.println("Invalid Position");

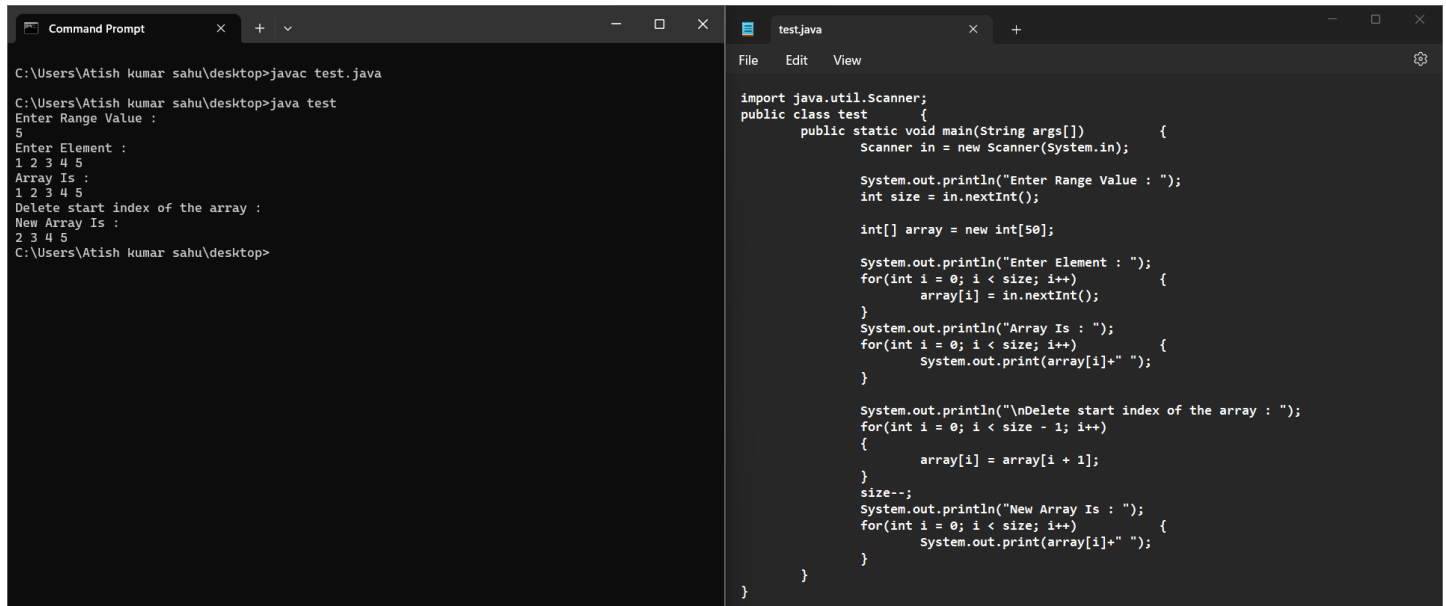
        for(int i = size - 1; i >= position - 1; i--)
        {
            array[i + 1] = array[i];
        }
        size++;
        array[position - 1] = num;

        System.out.println("New Array Is : ");
        for(int i = 0; i < size; i++)
        {
            System.out.print(array[i]+" ");
        }
    }
}
```

Question5: Deletion Of Element From Starting Index:

Take an array from user and delete the starting index of the array

Input: arr[1 2 3 4 5] output: arr[2,3,4,5]



The screenshot shows a Java program in a text editor and its execution in a command prompt. The program, named `test.java`, uses a `Scanner` to take input from the user. It prompts for a range value (5) and then elements (1 2 3 4 5). It prints the array as `1 2 3 4 5`. Then, it prompts for the start index to delete (2). The program shifts elements from index 2 onwards one position to the left and prints the new array as `2 3 4 5`.

```
import java.util.Scanner;
public class test
{
    public static void main(String args[])
    {
        Scanner in = new Scanner(System.in);

        System.out.println("Enter Range Value : ");
        int size = in.nextInt();

        int[] array = new int[50];

        System.out.println("Enter Element : ");
        for(int i = 0; i < size; i++)
        {
            array[i] = in.nextInt();
        }
        System.out.println("Array Is : ");
        for(int i = 0; i < size; i++)
        {
            System.out.print(array[i]+" ");
        }

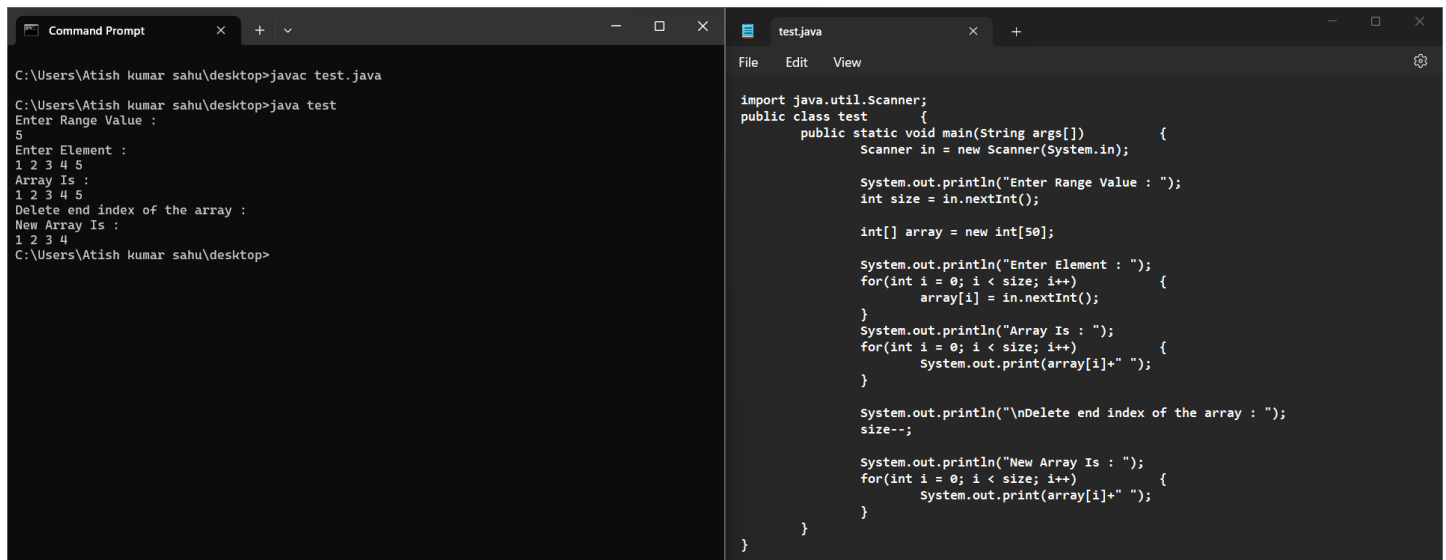
        System.out.println("\nDelete start index of the array : ");
        for(int i = 0; i < size - 1; i++)
        {
            array[i] = array[i + 1];
        }
        size--;
        System.out.println("New Array Is : ");
        for(int i = 0; i < size; i++)
        {
            System.out.print(array[i]+" ");
        }
    }
}
```

```
C:\Users\Atish kumar sahu\desktop>javac test.java
C:\Users\Atish kumar sahu\desktop>java test
Enter Range Value :
5
Enter Element :
1 2 3 4 5
Array Is :
1 2 3 4 5
Delete start index of the array :
New Array Is :
2 3 4 5
C:\Users\Atish kumar sahu\desktop>
```

Question6: Deletion Of Element From End Index

Take an array from user and delete the end index of the array

Input: arr[1 2 3 4 5] output: arr[1,2,3,4]



The screenshot shows a Java program in a text editor and its execution in a command prompt. The program, named `test.java`, uses a `Scanner` to take input from the user. It prompts for a range value (5) and then elements (1 2 3 4 5). It prints the array as `1 2 3 4 5`. Then, it prompts for the end index to delete (5). The program removes the element at the end index and prints the new array as `1 2 3 4`.

```
import java.util.Scanner;
public class test
{
    public static void main(String args[])
    {
        Scanner in = new Scanner(System.in);

        System.out.println("Enter Range Value : ");
        int size = in.nextInt();

        int[] array = new int[50];

        System.out.println("Enter Element : ");
        for(int i = 0; i < size; i++)
        {
            array[i] = in.nextInt();
        }
        System.out.println("Array Is : ");
        for(int i = 0; i < size; i++)
        {
            System.out.print(array[i]+" ");
        }

        System.out.println("\nDelete end index of the array : ");
        size--;

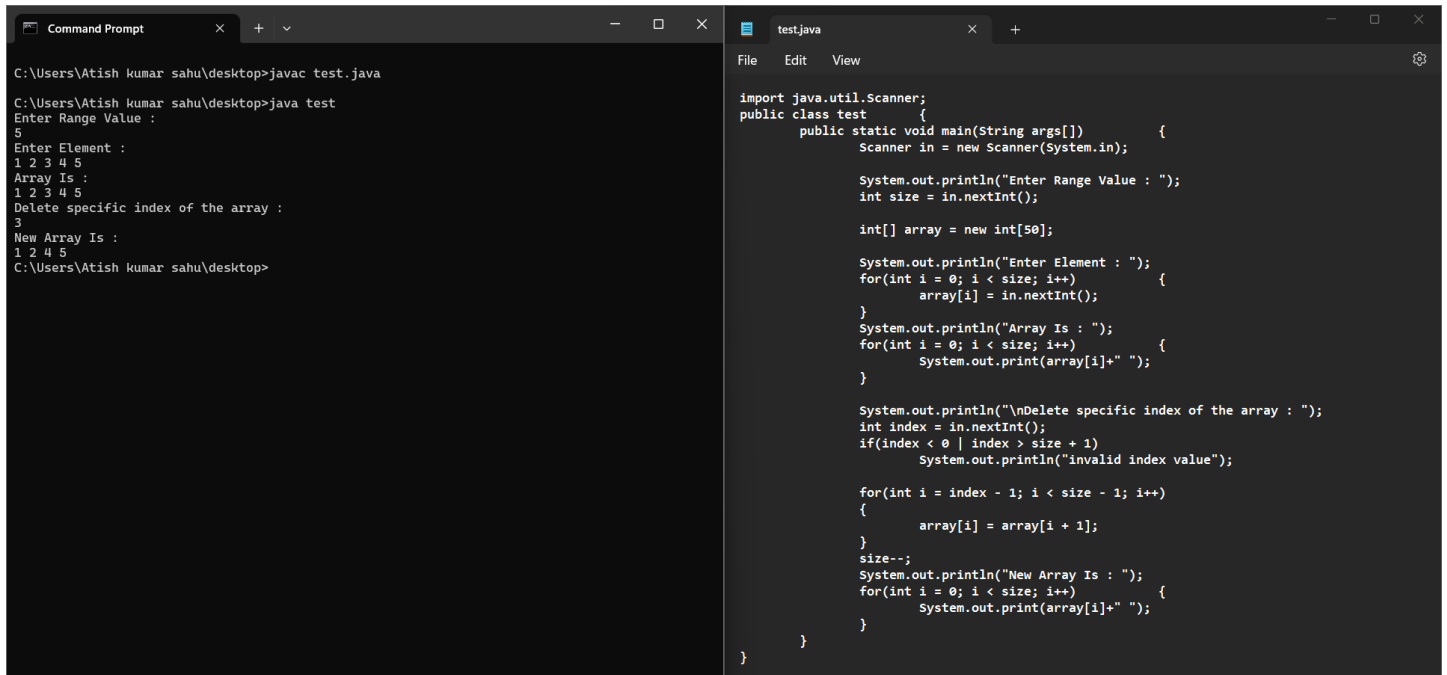
        System.out.println("New Array Is : ");
        for(int i = 0; i < size; i++)
        {
            System.out.print(array[i]+" ");
        }
    }
}
```

```
C:\Users\Atish kumar sahu\desktop>javac test.java
C:\Users\Atish kumar sahu\desktop>java test
Enter Range Value :
5
Enter Element :
1 2 3 4 5
Array Is :
1 2 3 4 5
Delete end index of the array :
New Array Is :
1 2 3 4
C:\Users\Atish kumar sahu\desktop>
```

Question7: Deletion Of Element From Specific Index

Take an array from user and delete the specific index of the array

Input: arr[1 2 3 4 5] index = 3 output: arr[1,2,4,5]



The screenshot shows a Java program in a text editor and its execution in a command prompt. The Java code defines a class 'test' with a 'main' method. It uses a 'Scanner' to take input for the range value (5) and elements (1 2 3 4 5). It then prompts for a specific index to delete (3). The code shifts elements from index 3 onwards one position to the left and prints the new array [1, 2, 4, 5].

```
import java.util.Scanner;
public class test
{
    public static void main(String args[])
    {
        Scanner in = new Scanner(System.in);

        System.out.println("Enter Range Value : ");
        int size = in.nextInt();

        int[] array = new int[50];

        System.out.println("Enter Element : ");
        for(int i = 0; i < size; i++)
        {
            array[i] = in.nextInt();
        }
        System.out.println("Array Is : ");
        for(int i = 0; i < size; i++)
        {
            System.out.print(array[i]+" ");
        }

        System.out.println("\nDelete specific index of the array : ");
        int index = in.nextInt();
        if(index < 0 || index > size + 1)
            System.out.println("Invalid index value");

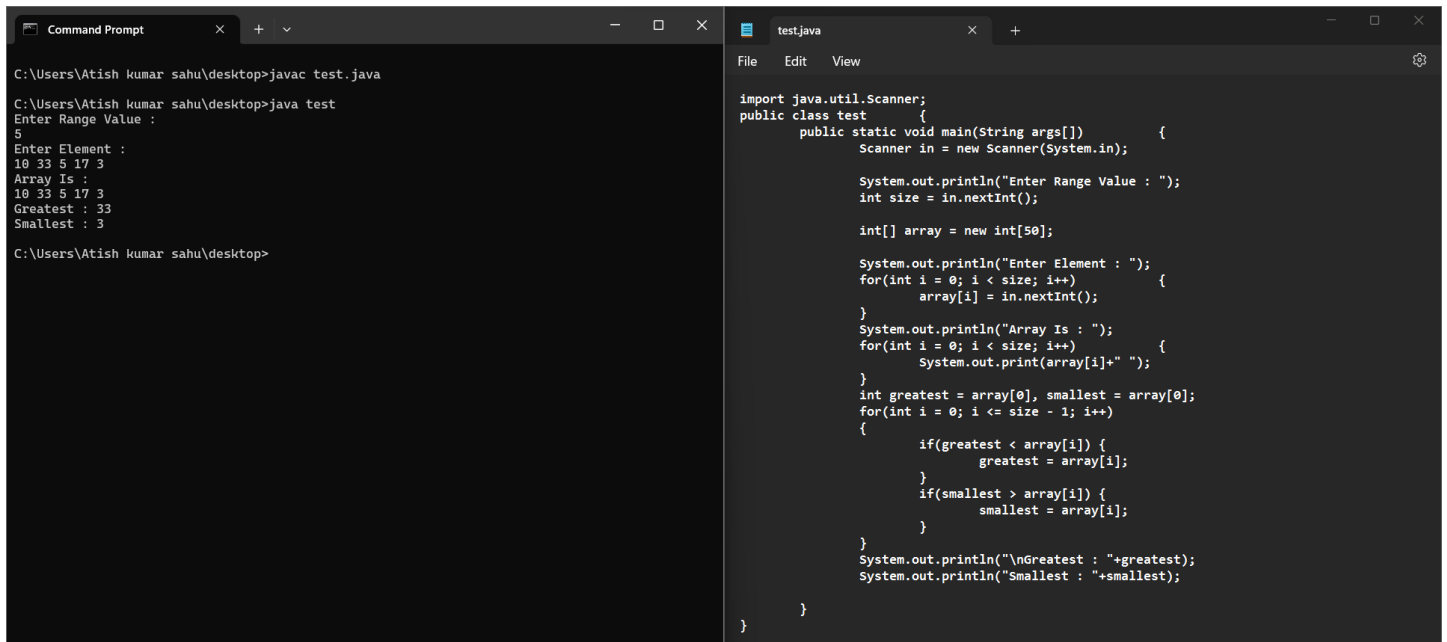
        for(int i = index - 1; i < size - 1; i++)
        {
            array[i] = array[i + 1];
        }
        size--;
        System.out.println("New Array Is : ");
        for(int i = 0; i < size; i++)
        {
            System.out.print(array[i]+" ");
        }
    }
}
```

```
C:\Users\Atish kumar sahu\desktop>javac test.java
C:\Users\Atish kumar sahu\desktop>java test
Enter Range Value :
5
Enter Element :
1 2 3 4 5
Array Is :
1 2 3 4 5
Delete specific index of the array :
3
New Array Is :
1 2 4 5
C:\Users\Atish kumar sahu\desktop>
```

Question 08: Greatest & Smallest Element Of Array

Take an array from user and find the greatest & smallest element from that array.

Input: arr[10 33 5 17 3] output: greatest: 33 smallest: 3



The screenshot shows a Java program in a text editor and its execution in a command prompt. The Java code defines a class 'test' with a 'main' method. It uses a 'Scanner' to take input for the range value (5) and elements (10 33 5 17 3). It then iterates through the array to find the greatest and smallest elements, printing the results: greatest: 33, smallest: 3.

```
import java.util.Scanner;
public class test
{
    public static void main(String args[])
    {
        Scanner in = new Scanner(System.in);

        System.out.println("Enter Range Value : ");
        int size = in.nextInt();

        int[] array = new int[50];

        System.out.println("Enter Element : ");
        for(int i = 0; i < size; i++)
        {
            array[i] = in.nextInt();
        }
        System.out.println("Array Is : ");
        for(int i = 0; i < size; i++)
        {
            System.out.print(array[i]+" ");
        }

        int greatest = array[0], smallest = array[0];
        for(int i = 0; i <= size - 1; i++)
        {
            if(greatest < array[i]) {
                greatest = array[i];
            }
            if(smallest > array[i]) {
                smallest = array[i];
            }
        }
        System.out.println("\nGreatest : "+greatest);
        System.out.println("Smallest : "+smallest);
    }
}
```

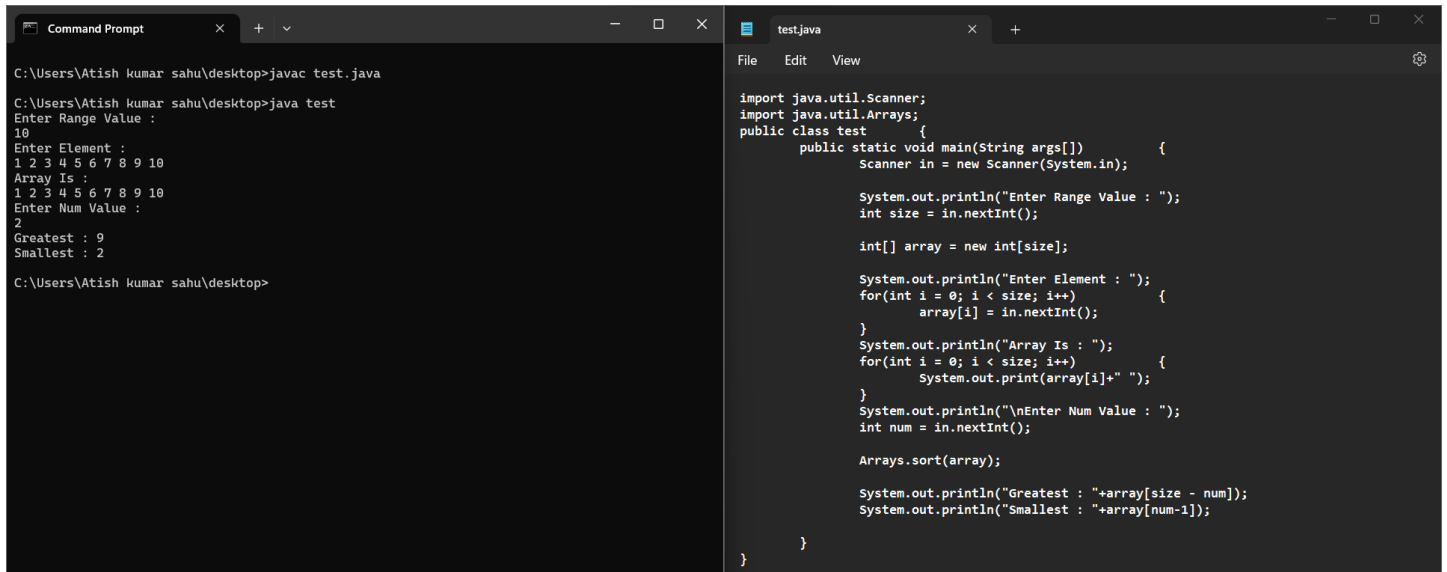
```
C:\Users\Atish kumar sahu\desktop>javac test.java
C:\Users\Atish kumar sahu\desktop>java test
Enter Range Value :
5
Enter Element :
10 33 5 17 3
Array Is :
10 33 5 17 3
Greatest : 33
Smallest : 3
C:\Users\Atish kumar sahu\desktop>
```


Question 09: Specific Greatest & Smallest Element In Array

Take an array from user and find the specific greatest and smallest element from that array.

Input: arr[1 2 3 4 5 6 7 8 9 10] num = 2

Output: Greatest = 9 Smallest = 2



The screenshot shows two windows. The left window is a Command Prompt showing the execution of a Java program. The user enters a range value of 10, then the elements 1 through 10. The program outputs the array and then asks for a number (2). It then outputs the greatest element (9) and the smallest element (2).

```
C:\Users\Atish kumar sahu\desktop>javac test.java
C:\Users\Atish kumar sahu\desktop>java test
Enter Range Value :
10
Enter Element :
1 2 3 4 5 6 7 8 9 10
Array Is :
1 2 3 4 5 6 7 8 9 10
Enter Num Value :
2
Greatest : 9
Smallest : 2
C:\Users\Atish kumar sahu\desktop>
```

The right window shows the Java source code for 'test.java'.

```
import java.util.Scanner;
import java.util.Arrays;
public class test
{
    public static void main(String args[])
    {
        Scanner in = new Scanner(System.in);

        System.out.println("Enter Range Value : ");
        int size = in.nextInt();

        int[] array = new int[size];

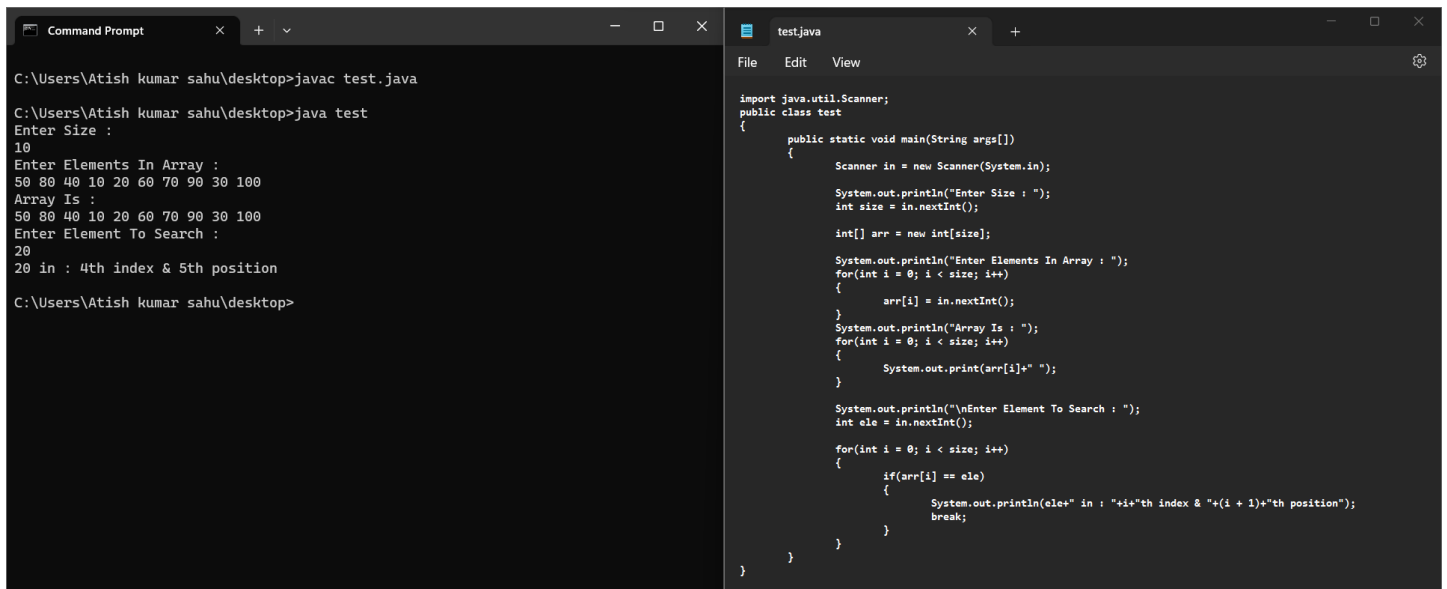
        System.out.println("Enter Element : ");
        for(int i = 0; i < size; i++)
        {
            array[i] = in.nextInt();
        }
        System.out.println("Array Is : ");
        for(int i = 0; i < size; i++)
        {
            System.out.print(array[i]+" ");
        }
        System.out.println("\nEnter Num Value : ");
        int num = in.nextInt();

        Arrays.sort(array);

        System.out.println("Greatest : "+array[size - num]);
        System.out.println("Smallest : "+array[num-1]);
    }
}
```

Question 10 Linear Search Algorithm

Write a program for linear search and find the given element in from the array and print the position and index value of that element.



The screenshot shows two windows. The left window is a Command Prompt showing the execution of a Java program. The user enters a size of 10, then the elements 50, 80, 40, 10, 20, 60, 70, 90, 30, 100. The program outputs the array and then asks for an element to search (20). It then outputs the position and index of the element (20 in : 4th index & 5th position).

```
C:\Users\Atish kumar sahu\desktop>javac test.java
C:\Users\Atish kumar sahu\desktop>java test
Enter Size :
10
Enter Elements In Array :
50 80 40 10 20 60 70 90 30 100
Array Is :
50 80 40 10 20 60 70 90 30 100
Enter Element To Search :
20
20 in : 4th index & 5th position
C:\Users\Atish kumar sahu\desktop>
```

The right window shows the Java source code for 'test.java'.

```
import java.util.Scanner;
public class test
{
    public static void main(String args[])
    {
        Scanner in = new Scanner(System.in);

        System.out.println("Enter Size : ");
        int size = in.nextInt();

        int[] arr = new int[size];

        System.out.println("Enter Elements In Array : ");
        for(int i = 0; i < size; i++)
        {
            arr[i] = in.nextInt();
        }
        System.out.println("Array Is : ");
        for(int i = 0; i < size; i++)
        {
            System.out.print(arr[i]+" ");
        }

        System.out.println("\nEnter Element To Search : ");
        int ele = in.nextInt();

        for(int i = 0; i < size; i++)
        {
            if(arr[i] == ele)
            {
                System.out.println(ele+" in : "+i+"th index & "+(i + 1)+"th position");
                break;
            }
        }
    }
}
```

Question 11

Binary Search Ascending Order

Write a program for the binary search and find the given element in from the array and print the position and index value of that element. Ascending order.

```
Command Prompt
C:\Users\Atish kumar sahu\desktop>javac test.java
C:\Users\Atish kumar sahu\desktop>java test
Enter Size :
8
Enter Elements In Array :
40 60 20 30 50 10 70 80
Sorted Array Is :
10 20 30 40 50 60 70 80
Enter Element To Search :
70
70 in : 6 index and 7 p osition

C:\Users\Atish kumar sahu\desktop>java test
Enter Size :
10
Enter Elements In Array :
80 20 10 90 30 40 70 60 50 100
Sorted Array Is :
10 20 30 40 50 60 70 80 90 100
Enter Element To Search :
40
40 in : 3 index and 4 p osition

C:\Users\Atish kumar sahu\desktop>

test.java
import java.util.Scanner;
import java.util.Arrays;
public class test
{
    public static void main(String args[])
    {
        Scanner in = new Scanner(System.in);

        System.out.println("Enter Size : ");
        int size = in.nextInt();

        int[] arr = new int[size];

        System.out.println("Enter Elements In Array : ");
        for(int i = 0; i < size; i++)
        {
            arr[i] = in.nextInt();
        }
        System.out.println("Sorted Array Is : ");
        Arrays.sort(arr);
        for(int i = 0; i < size; i++)
        {
            System.out.print(arr[i]+" ");
        }

        System.out.println("\nEnter Element To Search : ");
        int ele = in.nextInt(), last = 0, first = size - 1, mid = (last + first)/2;

        while(last <= first)
        {
            if(arr[mid] < ele)
                last = mid + 1;
            else if(arr[mid] == ele)
            {
                System.out.println(ele+" in : "+mid+" index and "+(mid + 1)+" p osition");
                break;
            }
            else
            {
                first = mid - 1;
                mid = (last + first) / 2;
            }
        }
        if(last > first)
            System.out.println(ele+" is not found in array");
    }
}
```

```
Command Prompt
C:\Users\Atish kumar sahu\desktop>javac test.java
C:\Users\Atish kumar sahu\desktop>java test
Enter Size Value :
10
Enter Array Elements :
10 20 30 40 50 60 70 80 90 100
Array Is :
A[0] : 10
A[1] : 20
A[2] : 30
A[3] : 40
A[4] : 50
A[5] : 60
A[6] : 70
A[7] : 80
A[8] : 90
A[9] : 100
Enter Value Of Search :
40
Output : 3 th index 4 th position

C:\Users\Atish kumar sahu\desktop>

test.java
import java.util.*;
public class test
{
    public static int function(int[] ar, int target)
    {
        int len = ar.length, last = 0, first = len - 1;
        while(last <= first)
        {
            int mid = last + (first - last) / 2;
            if(ar[mid] > target)
                first = mid - 1;
            else if(ar[mid] == target)
                return mid;
            else
                last = mid + 1;
        }
        return last;
    }
    public static void main(String args[])
    {
        Scanner in = new Scanner(System.in);
        System.out.println("Enter Size Value : ");
        int size = in.nextInt();

        int[] array = new int[size];
        System.out.println("Enter Array Elements : ");
        for(int i = 0; i < size; i++)
        {
            array[i] = in.nextInt();
        }
        System.out.println("Array Is : ");
        for(int i = 0; i < size; i++)
        {
            System.out.println("A["+i+"] : "+array[i]);
        }
        System.out.println("Enter Value Of Search : ");
        int target = in.nextInt();
        int output = function(array, target);
        System.out.println("Output : "+output+" th index "+(output + 1)+" th position");
    }
}
```

Question 12

Binary Search Descending Order

Binary Search with an array which elements are in descending order.

```
Command Prompt
C:\Users\Atish kumar sahu\desktop>javac test.java

C:\Users\Atish kumar sahu\desktop>java test
Enter Size Value :
10
Enter Elements :
100 90 80 70 60 50 40 30 20 10
Array Is :
A[0] : 100
A[1] : 90
A[2] : 80
A[3] : 70
A[4] : 60
A[5] : 50
A[6] : 40
A[7] : 30
A[8] : 20
A[9] : 10
Enter Value For Search :
20
Output : 8 th index 9 th position

C:\Users\Atish kumar sahu\desktop>
```

```
test.java
import java.util.*;
public class test
{
    public static int function(int[] ar, int num)
    {
        int len = ar.length,
            last = 0, first = len - 1;

        while(last <= first)
        {
            int mid = last + (first - last) / 2;
            if(ar[mid] < num)
                first = mid - 1;
            else if(ar[mid] == num)
                return mid;
            else
                last = mid + 1;
        }
        return last;
    }

    public static void main(String args[])
    {
        Scanner in = new Scanner(System.in);
        System.out.println("Enter Size Value : ");
        int size = in.nextInt();
        int[] array = new int[size];
        System.out.println("Enter Elements : ");
        for(int i = 0; i < size; i++)
        {
            array[i] = in.nextInt();
        }
        System.out.println("Array Is : ");
        for(int i = 0; i < size; i++)
        {
            System.out.println("A["+i+"] : "+array[i]);
        }
        System.out.println("Enter Value For Search : ");
        int target = in.nextInt();
        int output = function(array, target);
        System.out.println("Output : "+output+" th index "+(output + 1)+" th position");
    }
}
```

Question 13:

Sub Array In Java

```
Command Prompt
C:\Users\Atish kumar sahu\desktop>javac test.java

C:\Users\Atish kumar sahu\desktop>java test
Enter Size:
7
Enter Elements :
1 2 3 4 5 6 7
Sub Array :
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
1 2 3 4 5 6
1 2 3 4 5 6 7
2
2 3
2 3 4
2 3 4 5
2 3 4 5 6
2 3 4 5 6 7
3
3 4
3 4 5
3 4 5 6
3 4 5 6 7
4
4 5
4 5 6
4 5 6 7
5
5 6
5 6 7
6
6 7
7

C:\Users\Atish kumar sahu\desktop>
```

```
test.java
import java.util.Scanner;
public class test
{
    public static void main(String args[])
    {
        Scanner in = new Scanner(System.in);

        System.out.println("Enter Size: ");
        int size = in.nextInt();

        int[] array = new int[size];

        System.out.println("Enter Elements : ");
        for(int i = 0; i < size; i++)
        {
            array[i] = in.nextInt();
        }
        System.out.println("Sub Array : ");
        for(int i = 0; i < size; i++)
        {
            for(int j = i; j < size; j++)
            {
                for(int k = i; k <= j; k++)
                {
                    System.out.print(array[k]+" ");
                }
                System.out.println();
            }
        }
    }
}
```

Question 14:

Maximum sum in subarray: IP:[5 2 -4 -5 3 -1 2 3 1] OP: MAX: 8

Command Prompt

C:\Users\Atish kumar sahu\desktop>javac test.java
C:\Users\Atish kumar sahu\desktop>java test
Enter Size:
9
Enter Elements :
5 2 -4 -5 3 -1 2 3 1
Sub Array :
Max : 8
C:\Users\Atish kumar sahu\desktop>

test.java

File Edit View
import java.util.Scanner;
public class test
{
 public static void main(String args[])
 {
 Scanner in = new Scanner(System.in);

 System.out.println("Enter Size:");
 int size = in.nextInt();

 int[] array = new int[size];
 int count = 0, max = Integer.MIN_VALUE;
 System.out.println("Enter Elements : ");
 for(int i = 0; i < size; i++)
 {
 array[i] = in.nextInt();
 }
 System.out.println("Sub Array :");
 for(int i = 0; i < size; i++)
 {
 count += array[i];
 if(max < count)
 max = count;
 if(count < 0)
 count = 0;
 }
 System.out.println("Max : "+max);
 }
}

Question 15:

Frequency Of An Array.

C:\Users\Atish kumar sahu\desktop>javac test.java
C:\Users\Atish kumar sahu\desktop>java test
Enter Size :
10
Enter Array Values :
1 2 2 3 3 3 4 4 5 6
Array Elements Are :
1
2
2
3
3
3
4
4
5
6
Frequency Of Array Elements Are :
1---1
2---2
3---3
4---2
5---1
6---1
C:\Users\Atish kumar sahu\desktop>

test.java

File Edit View
import java.util.Scanner;
public class test
{
 public static void main(String args[])
 {
 Scanner in = new Scanner(System.in);
 System.out.println("Enter Size : ");
 int size = in.nextInt();
 int[] array = new int[size];

 System.out.println("Enter Array Values : ");
 for(int i = 0; i < size; i++) {
 array[i] = in.nextInt();
 }
 System.out.println("Array Elements Are : ");
 for(int i = 0; i < size; i++) {
 System.out.println(array[i]+" ");
 }
 int[] fr = new int[array.length];
 int visited = -1;
 for(int i = 0; i < array.length; i++) {
 int count = 1;
 for(int j = i + 1; j < array.length; j++) {
 if(array[i] == array[j]) {
 count++;
 fr[j] = visited;
 }
 }
 if(fr[i] != visited)
 fr[i] = count;
 }
 System.out.println("Frequency Of Array Elements Are : ");
 for(int i = 0; i < fr.length; i++) {
 if(fr[i] != visited)
 System.out.println(array[i]+"---"+fr[i]);
 }
 }
}

Question 16

Quick Sort Algorithm

```
test.java
File Edit View

import java.util.Scanner;
public class test
{
    public static int ascpartition(int[] ar, int lb, int ub)
    {
        int i = lb, j = ub;
        int pivot = ar[ub];

        while(i < j)
        {
            while(i < j && ar[i] <= pivot)
                i++;
            while(i < j && ar[j] >= pivot)
                j--;
            if(i < j)
            {
                int temp = ar[i];
                ar[i] = ar[j];
                ar[j] = temp;
            }
        }
        if(ar[i] > pivot)
        {
            int temp1 = ar[i];
            ar[i] = pivot;
            ar[ub] = temp1;
        }
        return i;
    }
}
```

```
public static void ascendingorder(int[] ar, int lb, int ub)
{
    if(lb < ub)
    {
        int pivot = ascpartition(ar, lb, ub);
        ascendingorder(ar, lb, pivot - 1);
        ascendingorder(ar, pivot + 1, ub);
    }
}
```

```
public static void descendingorder(int[] ar, int lb, int ub)
{
    if(lb < ub)
    {
        int pivot = descpartition(ar, lb, ub);
        descendingorder(ar, lb, pivot - 1);
        descendingorder(ar, pivot + 1, ub);
    }
}
```

```

public static int descpartition(int[] ar, int lb, int ub)
{
    int i = lb, j = ub;
    int pivot = ar[lb];

    while(i <= j)
    {
        while(i <= j && ar[i] >= pivot)
            i++;
        while(i <= j && ar[j] < pivot)
            j--;
        if(i < j)
        {
            int temp = ar[i];
            ar[i] = ar[j];
            ar[j] = temp;
        }
    }
    if(ar[j] > pivot)
    {
        int temp = ar[lb];
        ar[lb] = ar[j];
        ar[j] = temp;
    }
    return j;
}

```

Question 17

Bubble Sort Algorithm

```

public static void ascendingorder(int[] ar, int len)
{
    boolean flag;

    for(int i = 0; i < len - 1; i++)
    {
        flag = false;
        for(int j = 0; j < (len - 1 - i); j++)
        {
            if(ar[j] > ar[j + 1])
            {
                int temp = ar[j];
                ar[j] = ar[j + 1];
                ar[j + 1] = temp;

                flag = true;
            }
        }
        if(flag == false)
            break;
    }
}

```

```

public static void descendingorder(int[] ar, int len)
{
    boolean flag;

    for(int i = 0; i < len - 1; i++)
    {
        flag = false;
        for(int j = 0; j < (len - 1 - i); j++)
        {
            if(ar[j] < ar[j+1])
            {
                int temp = ar[j];
                ar[j] = ar[j + 1];
                ar[j + 1] = temp;

                flag = true;
            }
        }
        if(flag == false)
            break;
    }
}

```

Question 18

Insertion Sort Algorithm

```

System.out.println("Ascending Order Sort : ");
for(int i = 0; i < size; i++)
{
    int temp = array[i];
    int j = i - 1;
    while(j >= 0 && array[j] > temp)
    {
        array[j + 1] = array[j];
        j--;
    }
    array[j + 1] = temp;
}
for(int i = 0; i < size; i++)
{
    System.out.print(array[i]+" ");
}
System.out.println();

```

```

        System.out.println("Descending Order Sort : ");
        for(int i = 1; i < size; i++)
        {
            int temp = array[i];
            int j = i - 1;
            while(j >= 0 && array[j] < temp)
            {
                array[j + 1] = array[j];
                j--;
            }
            array[j + 1] = temp;
        }
        for(int i = 0; i < size; i++)
        {
            System.out.print(array[i]+" ");
        }
        System.out.println();
    }
}

```

Question 19

Selection Sort Algorithm

```

import java.util.Scanner;
public class Selection_Sort
{
    public static int[] AscendingOrder(int[] ar1)
    {
        int n = ar1.length;

        for(int i = 0; i < n - 1; i++)
        {
            int min = i;
            for(int j = i + 1; j < n; j++)
            {
                if(ar1[j] < ar1[min])
                {
                    min = j;
                }
            }
            if(min != i)
            {
                int temp = ar1[i];
                ar1[i] = ar1[min];
                ar1[min] = temp;
            }
        }
        return ar1;
    }
}

```



```
public static int[] DscendigOrder(int sz, int[] ar2)
{
    for(int i = 0; i < sz - 1; i++)
    {
        int max = i;
        for(int j = i + 1; j < sz; j++)
        {
            if(ar2[j] > ar2[max])
            {
                max = j;
            }
        }
        if(max != i)
        {
            int temp = ar2[i];
            ar2[i] = ar2[max];
            ar2[max] = temp;
        }
    }
    return ar2;
}
```

Question 20

Merge Sort Algorithm

```
public static void mergesorta(int[] a, int beg, int end)
{
    if(beg < end)
    {
        int mid = (beg + end) / 2;
        mergesorta(a, beg, mid);
        mergesorta(a, mid + 1, end);
        mergesortasc(a, beg, mid, end);
    }
}
```

```
public static void mergesortasc(int[] a, int beg, int mid, int end) {
    int i, j, k, n1 = mid - beg + 1, n2 = end - mid;

    int leftarray[] = new int[n1];
    int rightarray[] = new int[n2];

    for(i = 0; i < n1; i++)
        leftarray[i] = a[beg + i];
    for(j = 0; j < n2; j++)
        rightarray[j] = a[mid + 1 + j];

    i = 0; j = 0; k = beg;

    while(i < n1 && j < n2) {
        if(leftarray[i] <= rightarray[j]) {
            a[k] = leftarray[i];
            i++;
        } else {
            a[k] = rightarray[j];
            j++;
        }
        k++;
    }
    while(i < n1) {
        a[k] = leftarray[i];
        i++;
        k++;
    }
    while(j < n2) {
        a[k] = rightarray[j];
        j++;
        k++;
    }
}
```

```

public static void mergesortdesc(int[] a, int beg, int mid, int end) {
    int i, j, k, n1 = mid - beg + 1, n2 = end - mid;

    int leftarray[] = new int[n1];
    int rightarray[] = new int[n2];

    for(i = 0; i < n1; i++)
        leftarray[i] = a[beg + i];
    for(j = 0; j < n2; j++)
        rightarray[j] = a[mid + 1 + j];

    i = 0; j = 0; k = beg;

    while(i < n1 && j < n2) {
        if(leftarray[i] > rightarray[j]) {
            a[k] = leftarray[i];
            i++;
        }else {
            a[k] = rightarray[j];
            j++;
        }
        k++;
    }
    while(i < n1) {
        a[k] = leftarray[i];
        i++;
        k++;
    }
    while(j < n2) {
        a[k] = rightarray[j];
        j++;
        k++;
    }
}

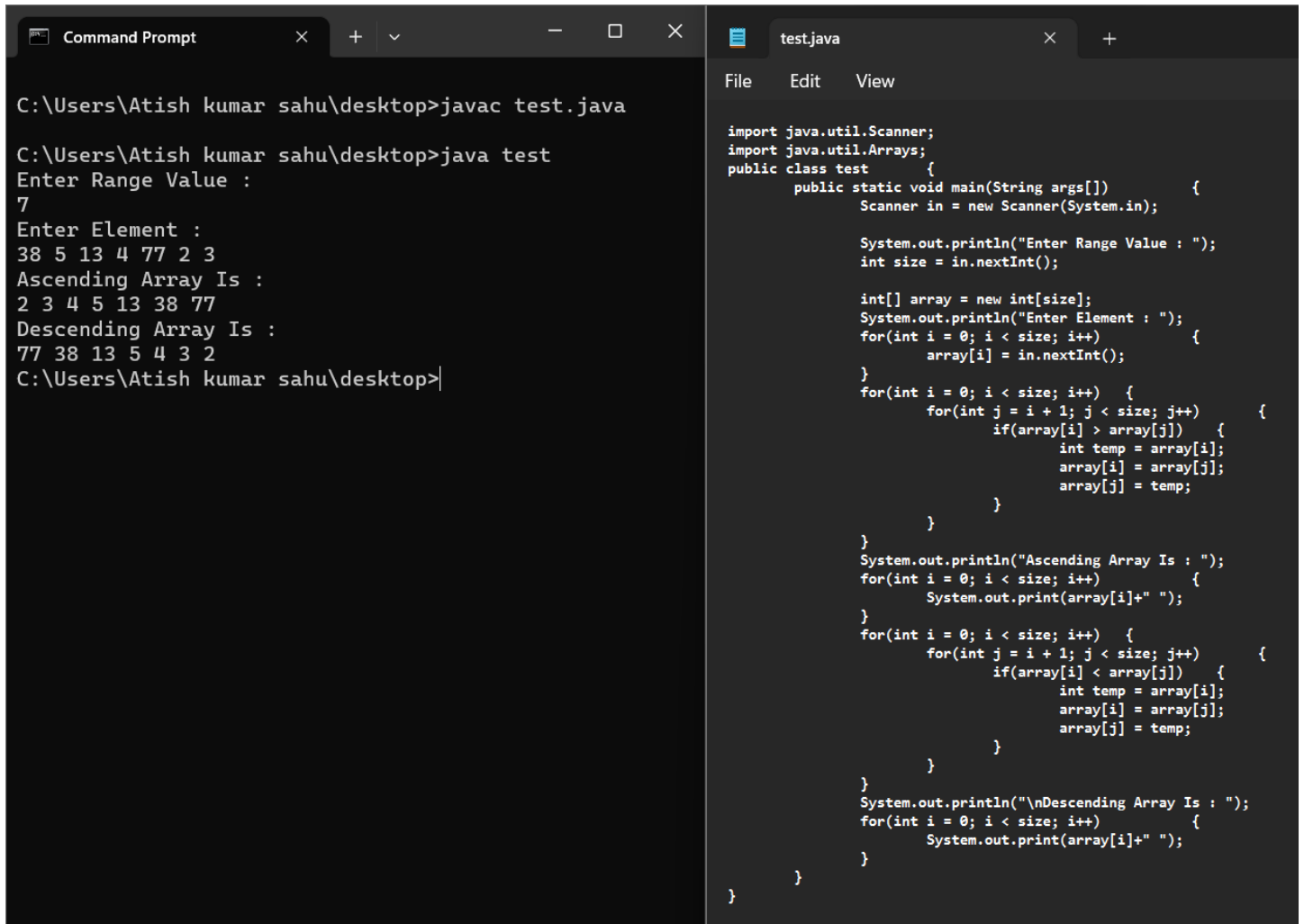
```

Question 21

Ascending & Descending Order Sort

Take an array from user and print the ascending and descending order of the array

Input: arr[10 40 30 20 50] output: arr[10,20,30,40,50] arr[50,40,30,20,10]



The image shows a screenshot of a Java program being executed in a Command Prompt and its source code in an IDE. The Command Prompt shows the user entering the range value (7) and the elements (38 5 13 4 77 2 3). The program then prints the ascending array (2 3 4 5 13 38 77) and the descending array (77 38 13 5 4 3 2). The IDE shows the source code for the 'test' class, which uses a Scanner to take input and implements bubble sort for both ascending and descending orders.

```
Command Prompt
C:\Users\Atish kumar sahu\desktop>javac test.java
C:\Users\Atish kumar sahu\desktop>java test
Enter Range Value :
7
Enter Element :
38 5 13 4 77 2 3
Ascending Array Is :
2 3 4 5 13 38 77
Descending Array Is :
77 38 13 5 4 3 2
C:\Users\Atish kumar sahu\desktop>

test.java
import java.util.Scanner;
import java.util.Arrays;
public class test {
    public static void main(String args[]) {
        Scanner in = new Scanner(System.in);

        System.out.println("Enter Range Value : ");
        int size = in.nextInt();

        int[] array = new int[size];
        System.out.println("Enter Element : ");
        for(int i = 0; i < size; i++) {
            array[i] = in.nextInt();
        }
        for(int i = 0; i < size; i++) {
            for(int j = i + 1; j < size; j++) {
                if(array[i] > array[j]) {
                    int temp = array[i];
                    array[i] = array[j];
                    array[j] = temp;
                }
            }
        }
        System.out.println("Ascending Array Is : ");
        for(int i = 0; i < size; i++) {
            System.out.print(array[i]+" ");
        }
        for(int i = 0; i < size; i++) {
            for(int j = i + 1; j < size; j++) {
                if(array[i] < array[j]) {
                    int temp = array[i];
                    array[i] = array[j];
                    array[j] = temp;
                }
            }
        }
        System.out.println("\nDescending Array Is : ");
        for(int i = 0; i < size; i++) {
            System.out.print(array[i]+" ");
        }
    }
}
```

Stack Implementation In Java:

```
import java.util.*;
public class test
{
    static int maxsize;
    static int[] array;
    static int top;

    test(int size){
        maxsize = size;
        array = new int[maxsize];
        top = -1;
    }
```

```
    public static void push()
    {
        Scanner in = new Scanner(System.in);
        System.out.println("Enter A Value To Push : ");
        int num = in.nextInt();
        if(top == maxsize - 1)
            System.out.println("Overflow!");
        else{
            top++;
            array[top] = num;
        }
    }
```

```
    public static void pop()
    {
        Scanner in = new Scanner(System.in);
        System.out.println("Enter A Value To Pop : ");
        int item = in.nextInt();
        if(top == -1)
            System.out.println("Underflow!");
        else{
            item = array[top];
            top--;
            System.out.println("Pop Value : "+item);
        }
    }

    public static void peek()
    {
        if(top == -1)
            System.out.println("UnderFlow!!");
        else
            System.out.println("Top Value : "+array[top]);
    }
```

```
    public static void isEmpty()
    {
        if(top == -1)
            System.out.println("Stack Is Empty!!!");
        else
            System.out.println("Stack Is Not Empty!!!");
    }
```

```

public static void TDdisplay()
{
    if(top == -1)
        System.out.println("No Element For Display!!!!");
    else{
        for(int i = top; i >= 0; i--)
        {
            System.out.println("a["+i+"] : "+array[i]);
        }
        System.out.println();
    }
}

```

```

public static void DTdisplay()
{
    if(top == -1)
        System.out.println("No Element For Display!!!!");
    else{
        for(int i = 0; i <= top; i++)
        {
            System.out.println("a["+i+"] : "+array[i]);
        }
        System.out.println();
    }
}

```

```

public static void main(String args[])
{
    Scanner in = new Scanner(System.in);
    System.out.println("Enter Size : ");
    int n = in.nextInt();
    test t = new test(n);
    int choice;
    do{
        System.out.println("Enter Choice : ");
        choice = in.nextInt();
        switch(choice)
        {
            case 1 :
                push();
                break;
            case 2 :
                pop();
                break;
            case 3 :
                peek();
                break;
            case 4 :
                isEmpty();
                break;
            case 5 :
                TDdisplay();
                break;
            case 6 :
                DTdisplay();
                break;
            default :
                System.out.println("Invalid Choice!!!");
                break;
        }
    }while(choice != 0);
}
}

```

Queue Implementation In Java:

```
import java.util.*;
public class test
{
    public static int front, rear, size, capacity;
    public static int[] array;

    test(int s)
    {
        this.capacity = s;
        front = this.size = 0;
        rear = capacity - 1;
        array = new int[this.capacity];
    }

    boolean isfull(test t)
    {
        return (t.size == t.capacity);
    }
    boolean isempty(test t)
    {
        return (t.size == 0);
    }

    public void full()
    {
        System.out.println("Queue Is Full : "+(size == capacity));
    }
    public void empty()
    {
        System.out.println("Queue Is Empty : "+(size == 0));
    }
    public void enqueue(int x)
    {
        if(isfull(this))
            return;
        this.rear = (this.rear + 1) % this.capacity;
        this.array[this.rear] = x;
        this.size = this.size + 1;
    }
    public void dequeue()
    {
        if(isempty(this))
            System.out.println("Queue Is Empty");
        int item = this.array[this.front];
        System.out.println("Deleted Value : "+item);
        this.front = (this.front + 1) % this.capacity;
        this.size = this.size - 1;
    }
}
```

```

public void front()
{
    if(isempty(this))
        System.out.println("Queue Is Empty");
    System.out.println("Front Value : "+this.array[this.front]);
}
public void rear()
{
    if(isempty(this))
        System.out.println("Queue Is Empty");
    System.out.println("Rear Value : "+this.array[this.rear]);
}

public void display()
{
    if(isempty(this))
        System.out.println("Queue Is Empty");
    System.out.println("Queue Is : ");
    int i = front, count = 0;
    while(count < size){
        System.out.println("a["+i+"] : "+array[i]);
        i = (i + 1) % capacity;
        count++;
    }
}

```

```

public static void main(String args[]) {
    Scanner in = new Scanner(System.in);
    System.out.println("Enter Size : ");
    int n = in.nextInt();
    test t = new test(n);
    int choice;
    do{
        System.out.println("Enter Choice : ");
        choice = in.nextInt();
        switch(choice){
            case 1 :
                System.out.println("Enter Value : ");
                int value = in.nextInt();
                t.enqueue(value);
                break;
            case 2 :
                t.dequeue();
                break;
            case 3 :
                t.full();
                break;
            case 4 :
                t.empty();
                break;
            case 5 :
                t.front();
                break;
            case 6 :
                t.rear();
                break;
            case 7 :
                t.display();
                break;
            default:
                System.out.println("Invalid Value : ");
                break;
        }
    }while(choice != 0);
}
}

```


Circular Queue:

```
import java.util.*;
public class test
{
    public static int front, rear, size;
    public static int[] array;

    test(int s)
    {
        size = s + 1;
        array = new int[size];
        front = rear = 0;
    }

    public static void enqueue()
    {
        Scanner on = new Scanner(System.in);
        System.out.println("Enter Element For Enqueue : ");
        int element = on.nextInt();

        if((rear + 1) % size == front)
        {
            System.out.println("Circular Queue Is Full. No Insertion Operation.");
        }
        else
        {
            rear = (rear + 1) % size;
            array[rear] = element;
            System.out.println("Inserted Element : "+element);
        }
    }
}
```

```
public static void dequeue()
{
    if(front == rear)
    {
        System.out.println("Circular Queue Is Empty.");
    }
    else
    {
        front = (front + 1) % size;
        int remove = array[front];
        System.out.println("Deleted Element : "+remove);
    }
}
```

```

public static void isfull()
{
    if((rear + 1) % size == front)
    {
        System.out.println("Circular Queue Is Full. ");
    }
}
public static void isempty()
{
    if(front == rear)
    {
        System.out.println("Circular Queue Is Empty.");
    }
}

```

```

public static void frontale()
{
    System.out.println("Front Element Of Circular Queue Is : "+array[(front + 1) % size]);
}

public static void rearele()
{
    System.out.println("Rear Element Of Circular Queue Is : "+array[rear]);
}

public static void display()
{
    if(front == rear)
    {
        System.out.println("Circular Queue Is Empty.");
    }
    else
    {
        System.out.println("Circular Queue IS : ");
        int current = (front + 1) % size;
        while(current != (rear + 1) % size)
        {
            System.out.print(array[current]+" ");
            current = (current + 1) % size;
        }
        System.out.println();
    }
}

```

```

public static void nextele()
{
    if(front == rear)
    {
        System.out.println("Circular Queue Is Empty.");
    }
    else
    {
        System.out.println("Current Element : "+array[rear]);
        int current = (rear + 1) % size;
        int val = array[(front + 1) % size];
        System.out.println("Next Element : "+val);
    }
}

```

```

public static void main(String args[]) {
    Scanner in = new Scanner(System.in);
    System.out.println("Enter Size Value : ");
    int n = in.nextInt();
    test t = new test(n);
    int choice;
    do{
        System.out.println("Enter Choice For Operation : ");
        choice = in.nextInt();
        switch(choice) {
            case 1 :
                enqueue();
                break;
            case 2 :
                dequeue();
                break;
            case 3 :
                isfull();
                break;
            case 4 :
                isempty();
                break;
            case 5 :
                fronte();
                break;
            case 6 :
                reare();
                break;
            case 7 :
                display();
                break;
            case 8 :
                nextele();
                break;
        }
    }while(choice != 0);
}
}

```

Double Ended Queue:

```
import java.util.*;
public class test
{
    public static int front, rear, size;
    public static int[] array;

    test(int s)
    {
        size = s + 1;
        array = new int[size];
        front = rear = -1;
    }

    public static void insert_front()
    {
        Scanner on = new Scanner(System.in);
        System.out.println("Enter Element For Insert : ");
        int element = on.nextInt();
        if((front == 0 && rear == size - 1) || (front == rear + 1))
        {
            System.out.println("Overflow Situation.");
        }
        else if((front == -1) && (rear == -1))
        {
            front = rear = 0;
            array[front] = element;
        }
        else if(front == 0)
        {
            front = size - 1;
            array[front] = element;
        }
        else
        {
            front = front - 1;
            array[front] = element;
        }
        System.out.println("Inserted Element At Front Is : "+element);
    }
}
```

```
public static void insert_rear()
{
    Scanner nn = new Scanner(System.in);
    System.out.println("Enter Element For Insert : ");
    int element = nn.nextInt();
    if((front == 0 && rear == size - 1) || (front == rear + 1))
    {
        System.out.println("Overflow Situation.");
    }
    else if((front == -1) && (rear == -1))
    {
        rear = 0;
        array[rear] = element;
    }
    else if(rear == size - 1)
    {
        rear = 0;
        array[rear] = element;
    }
    else
    {
        rear++;
        array[rear] = element;
    }
    System.out.println("Inserted Element At Rear Is : "+element);
}
```

```

}
public static void fronteLe()
{
    if((front == -1) && (rear == -1))
    {
        System.out.println("Dequeue Is Empty");
    }
    else
    {
        System.out.println("Front Element Of Dequeue : "+array[front]);
    }
}
public static void reareLe()
{
    if((front == -1) && (rear == -1))
    {
        System.out.println("Dequeue Is Empty");
    }
    else
    {
        System.out.println("Rear Element Of Dequeue : "+array[rear]);
    }
}
}

```

```

public static void delete_front()
{
    if((front == -1) && (rear == -1))
    {
        System.out.println("Dequeue Is Empty");
    }
    else if(front == rear)
    {
        System.out.println("Deleted Element Is : "+array[front]);
        front = -1;
        rear = -1;
    }
    else if(front == (size - 1))
    {
        System.out.println("Deleted Element Is : "+array[front]);
        front = 0;
    }
    else
    {
        System.out.println("Deleted Element Is : "+array[front]);
        front = front + 1;
    }
}
}

```

```

}
public static void display()
{
    int i = front;
    System.out.println("Elements are : ");
    while(i != rear)
    {
        System.out.print(array[i]+" ");
        i = (i + 1) % size;
    }
    System.out.println(array[rear]);
}
}

```

```

public static void delete_rear()
{
    if((front == -1) && (rear == -1))
    {
        System.out.println("Dequeue Is Empty");
    }
    else if(front == rear)
    {
        System.out.println("Deleted Element Is : "+array[rear]);
        front = -1;
        rear = -1;
    }
    else if(rear == 0)
    {
        System.out.println("Deleted Element Is : "+array[rear]);
        rear = size - 1;
    }
    else
    {
        System.out.println("Deleted Element Is : "+array[rear]);
        rear = rear - 1;
    }
}
}

```

```

public static void main(String args[]) {
    Scanner in = new Scanner(System.in);
    System.out.println("Enter Size Value : ");
    int n = in.nextInt();
    test t = new test(n);
    int choice;
    do{
        System.out.println("Enter Choice For Operation : ");
        choice = in.nextInt();
        switch(choice) {
            case 1 :
                insert_front();
                break;
            case 2 :
                insert_rear();
                break;
            case 3 :
                delete_front();
                break;
            case 4 :
                delete_rear();
                break;
            case 5 :
                frontele();
                break;
            case 6 :
                rearele();
                break;
            case 7 :
                display();
                break;
        }
    }while(choice != 0);
}
}
}

```

Singly Linked List:

```
import java.util.Scanner;
class Node {
    int data;
    Node next;

    Node(int data) {
        this.data = data;
        this.next = null;
    }
}
```

```
class LinkedList {
    private Node head;

    LinkedList() {
        head = null;
    }
    public void add(int data) {
        Node newNode = new Node(data);
        if (head == null) {
            head = newNode;
        } else {
            Node current = head;
            while (current.next != null) {
                current = current.next;
            }
            current.next = newNode;
        }
    }
}
```

```
public void delete(int data) {
    if (head == null) {
        System.out.println("List is empty. Nothing to delete.");
        return;
    }
    if (head.data == data) {
        head = head.next;
        return;
    }
    Node current = head;
    Node prev = null;
    while (current != null && current.data != data) {
        prev = current;
        current = current.next;
    }
    if (current == null) {
        System.out.println("Element not found in the list.");
    } else {
        prev.next = current.next;
    }
}
```

```

public void display() {
    Node current = head;
    if (current == null) {
        System.out.println("List is empty.");
        return;
    }
    System.out.print("Linked List: ");
    while (current != null) {
        System.out.print(current.data + " -> ");
        current = current.next;
    }
    System.out.println("null");
}
}

```

```

public class test {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        LinkedList linkedList = new LinkedList();

        System.out.print("Enter the size of the linked list: ");
        int size = scanner.nextInt();

        for (int i = 0; i < size; i++) {
            System.out.print("Enter element " + (i + 1) + ": ");
            int element = scanner.nextInt();
            linkedList.add(element);
        }

        linkedList.display();

        System.out.print("Enter an element to delete: ");
        int elementToDelete = scanner.nextInt();
        linkedList.delete(elementToDelete);

        linkedList.display();

        scanner.close();
    }
}

```



```
C:\Users\Atish kumar sahu\desktop>javac test.java

C:\Users\Atish kumar sahu\desktop>java test
Enter the size of the linked list: 10
Enter element 1: 10
Enter element 2: 20
Enter element 3: 30
Enter element 4: 40
Enter element 5: 50
Enter element 6: 60
Enter element 7: 70
Enter element 8: 80
Enter element 9: 90
Enter element 10: 100
Linked List: 10 -> 20 -> 30 -> 40 -> 50 -> 60 -> 70 -> 80 -> 90 -> 100 -> null
Enter an element to delete: 50
Linked List: 10 -> 20 -> 30 -> 40 -> 60 -> 70 -> 80 -> 90 -> 100 -> null

C:\Users\Atish kumar sahu\desktop>
```

Doubly Linked List:

```
import java.util.Scanner;
```

```
class Node {
    int data;
    Node next;
    Node prev;

    Node(int data) {
        this.data = data;
        this.next = null;
        this.prev = null;
    }
}
```

```
class DoublyLinkedList {
    private Node head;
    private Node tail;

    DoublyLinkedList() {
        head = null;
        tail = null;
    }

    public void add(int data) {
        Node newNode = new Node(data);
        if (head == null) {
            head = newNode;
            tail = newNode;
        } else {
            tail.next = newNode;
            newNode.prev = tail;
            tail = newNode;
        }
    }
}
```

```

public void delete(int data) {
    if (head == null) {
        System.out.println("List is empty. Nothing to delete.");
        return;
    }

    Node current = head;

    while (current != null && current.data != data) {
        current = current.next;
    }

    if (current == null) {
        System.out.println("Element not found in the list.");
        return;
    }

    if (current == head) {
        head = head.next;
        if (head != null) {
            head.prev = null;
        }
    } else if (current == tail) {
        tail = tail.prev;
        tail.next = null;
    } else {
        current.prev.next = current.next;
        current.next.prev = current.prev;
    }
}

```

```

public void display() {
    Node current = head;
    if (current == null) {
        System.out.println("List is empty.");
        return;
    }
    System.out.print("Doubly Linked List (forward): ");
    while (current != null) {
        System.out.print(current.data + " <-> ");
        current = current.next;
    }
    System.out.println("null");

    current = tail;
    System.out.print("Doubly Linked List (backward): ");
    while (current != null) {
        System.out.print(current.data + " <-> ");
        current = current.prev;
    }
    System.out.println("null");
}
}

```

```

public class test {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        DoublyLinkedList doublyLinkedList = new DoublyLinkedList();

        System.out.print("Enter the size of the doubly linked list: ");
        int size = scanner.nextInt();

        for (int i = 0; i < size; i++) {
            System.out.print("Enter element " + (i + 1) + ": ");
            int element = scanner.nextInt();
            doublyLinkedList.add(element);
        }

        doublyLinkedList.display();

        System.out.print("Enter an element to delete: ");
        int elementToDelete = scanner.nextInt();
        doublyLinkedList.delete(elementToDelete);

        doublyLinkedList.display();

        scanner.close();
    }
}

```

```
C:\Users\Atish kumar sahu\desktop>javac test.java
```

```
C:\Users\Atish kumar sahu\desktop>java test
```

```
Enter the size of the doubly linked list: 5
```

```
Enter element 1: 10
```

```
Enter element 2: 20
```

```
Enter element 3: 30
```

```
Enter element 4: 40
```

```
Enter element 5: 50
```

```
Doubly Linked List (forward): 10 <--> 20 <--> 30 <--> 40 <--> 50 <--> null
```

```
Doubly Linked List (backward): 50 <--> 40 <--> 30 <--> 20 <--> 10 <--> null
```

```
Enter an element to delete: 20
```

```
Doubly Linked List (forward): 10 <--> 30 <--> 40 <--> 50 <--> null
```

```
Doubly Linked List (backward): 50 <--> 40 <--> 30 <--> 10 <--> null
```

```
C:\Users\Atish kumar sahu\desktop>
```

Circular Singly Linked List:

```
import java.util.Scanner;
```

```
class Node {  
    int data;  
    Node next;  
  
    Node(int data) {  
        this.data = data;  
        this.next = null;  
    }  
}
```

```
class CircularSinglyLinkedList {  
    private Node head;  
  
    CircularSinglyLinkedList() {  
        head = null;  
    }  
  
    public void add(int data) {  
        Node newNode = new Node(data);  
        if (head == null) {  
            head = newNode;  
            newNode.next = head; // Make it circular  
        } else {  
            Node current = head;  
            while (current.next != head) {  
                current = current.next;  
            }  
            current.next = newNode;  
            newNode.next = head; // Make it circular  
        }  
    }  
}
```

```

public void delete(int data) {
    if (head == null) {
        System.out.println("List is empty. Nothing to delete.");
        return;
    }

    Node current = head;
    Node prev = null;

    // Find the node to delete
    while (current.data != data) {
        prev = current;
        current = current.next;

        if (current == head) {
            System.out.println("Element not found in the list.");
            return;
        }
    }

    if (current == head) {
        // Find the last node in the list
        Node last = head;
        while (last.next != head) {
            last = last.next;
        }

        head = head.next;
        last.next = head;
    } else {
        prev.next = current.next;
    }
}

```

```

public void display() {
    if (head == null) {
        System.out.println("List is empty.");
        return;
    }

    Node current = head;
    System.out.print("Circular Singly Linked List: ");
    do {
        System.out.print(current.data + " -> ");
        current = current.next;
    } while (current != head);
    System.out.println("(head)");
}
}

```

```

public class test {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        CircularSinglyLinkedList circularList = new CircularSinglyLinkedList();

        System.out.print("Enter the size of the circular singly linked list: ");
        int size = scanner.nextInt();

        for (int i = 0; i < size; i++) {
            System.out.print("Enter element " + (i + 1) + ": ");
            int element = scanner.nextInt();
            circularList.add(element);
        }

        circularList.display();

        System.out.print("Enter an element to delete: ");
        int elementToDelete = scanner.nextInt();
        circularList.delete(elementToDelete);

        circularList.display();

        scanner.close();
    }
}

```

```
C:\Users\Atish kumar sahu\desktop>javac test.java
```

```
C:\Users\Atish kumar sahu\desktop>java test
```

```
Enter the size of the circular singly linked list: 8
```

```
Enter element 1: 10
```

```
Enter element 2: 20
```

```
Enter element 3: 30
```

```
Enter element 4: 40
```

```
Enter element 5: 50
```

```
Enter element 6: 60
```

```
Enter element 7: 70
```

```
Enter element 8: 80
```

```
Circular Singly Linked List: 10 -> 20 -> 30 -> 40 -> 50 -> 60 -> 70 -> 80 -> (head)
```

```
Enter an element to delete: 80
```

```
Circular Singly Linked List: 10 -> 20 -> 30 -> 40 -> 50 -> 60 -> 70 -> (head)
```

```
C:\Users\Atish kumar sahu\desktop>
```

Circular doubly linked list:

```
import java.util.Scanner;

class Node {
    int data;
    Node next;
    Node prev;

    Node(int data) {
        this.data = data;
        this.next = this.prev = null;
    }
}
```

```
class CircularDoublyLinkedList {
    private Node head;

    CircularDoublyLinkedList() {
        head = null;
    }

    // Add an element to the Circular Doubly Linked List
    public void add(int data) {
        Node newNode = new Node(data);
        if (head == null) {
            head = newNode;
            head.next = head;
            head.prev = head;
        } else {
            Node tail = head.prev;
            tail.next = newNode;
            newNode.prev = tail;
            newNode.next = head;
            head.prev = newNode;
        }
    }
}
```

```

// Delete an element from the Circular Doubly Linked List
public void delete(int data) {
    if (head == null) {
        System.out.println("List is empty. Nothing to delete.");
        return;
    }

    Node current = head;

    do {
        if (current.data == data) {
            if (current == head && current.next == head) {
                head = null;
            } else {
                Node prevNode = current.prev;
                Node nextNode = current.next;
                prevNode.next = nextNode;
                nextNode.prev = prevNode;
                if (current == head) {
                    head = nextNode;
                }
            }
            return;
        }
        current = current.next;
    } while (current != head);

    System.out.println("Element not found in the list.");
}

```

```

// Display the Circular Doubly Linked List
public void display() {
    if (head == null) {
        System.out.println("List is empty.");
        return;
    }

    Node current = head;

    do {
        System.out.print(current.data + " <-> ");
        current = current.next;
    } while (current != head);

    System.out.println("(head)");
}

```



```

public class test {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        CircularDoublyLinkedList circularList = new CircularDoublyLinkedList();

        System.out.print("Enter the size of the Circular Doubly Linked List: ");
        int size = scanner.nextInt();

        for (int i = 0; i < size; i++) {
            System.out.print("Enter element " + (i + 1) + ": ");
            int element = scanner.nextInt();
            circularList.add(element);
        }

        circularList.display();

        System.out.print("Enter an element to delete: ");
        int elementToDelete = scanner.nextInt();
        circularList.delete(elementToDelete);

        circularList.display();

        scanner.close();
    }
}

```

```
C:\Users\Atish kumar sahu\desktop>javac test.java
```

```
C:\Users\Atish kumar sahu\desktop>java test
```

```
Enter the size of the Circular Doubly Linked List: 10
```

```
Enter element 1: 10
```

```
Enter element 2: 20
```

```
Enter element 3: 30
```

```
Enter element 4: 40
```

```
Enter element 5: 50
```

```
Enter element 6: 60
```

```
Enter element 7: 70
```

```
Enter element 8: 80
```

```
Enter element 9: 90
```

```
Enter element 10: 100
```

```
10 <-> 20 <-> 30 <-> 40 <-> 50 <-> 60 <-> 70 <-> 80 <-> 90 <-> 100 <-> (head)
```

```
Enter an element to delete: 50
```

```
10 <-> 20 <-> 30 <-> 40 <-> 60 <-> 70 <-> 80 <-> 90 <-> 100 <-> (head)
```

```
C:\Users\Atish kumar sahu\desktop>
```