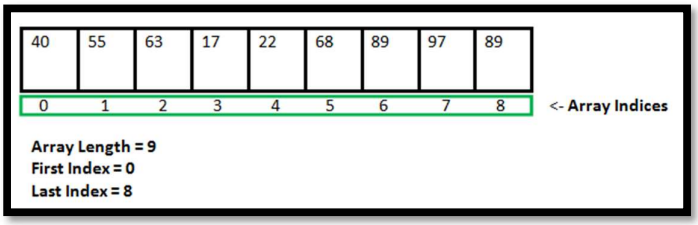
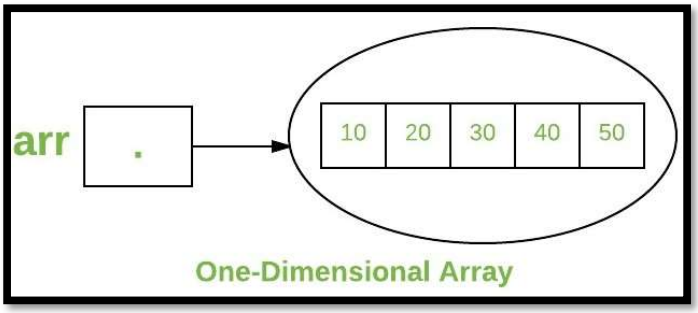
Array

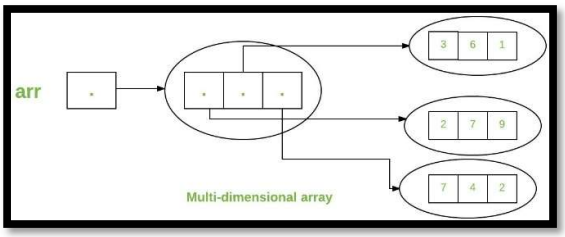
Array 1 Dimensi






```
package latarray;
//Latihan Array 1 dimensi
public class latihanar1 {
    public static void main(String[] args) {
        // TODO code application logic here
        int[] arr; // deklarasi array type data integer
        arr = new int[5]; //alokasi memori 5
        arr[0] = 10; //elemen pertama array
        arr[1] = 20; //elemen kedua array
        arr[2] = 30;
        arr[3] = 40;
        arr[4] = 50;
        //menampilkan isi array
        for (int i = 0; i < arr.length; i++)
         System.out.println("Elemen di setiap index " + i + " : "+ arr[i]);
    }
}
```

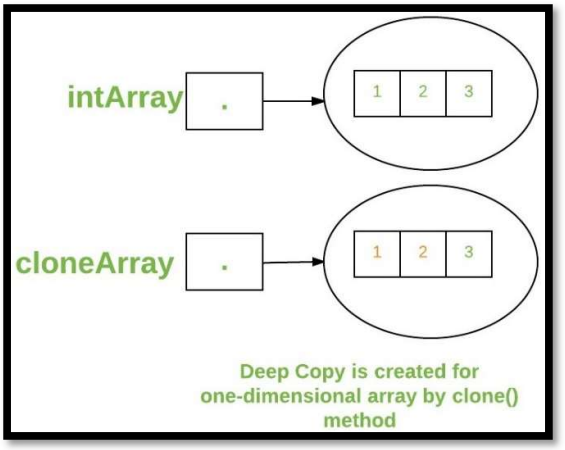
Multidimensional Arrays

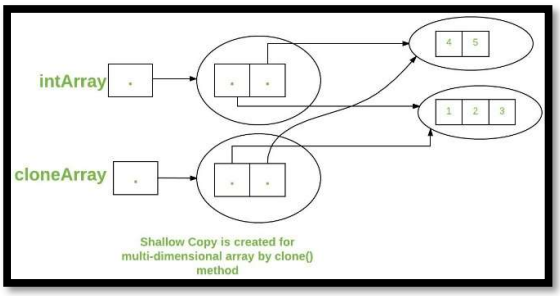



```
package latarray;
//Latihan array multi dimensi
public class latihanar2 {

    public static void main(String[] args) {
        // TODO code application logic here
        int arr[][] = { {2,7,9},{3,6,1},{7,4,2} }; //deklarasi 2 dimensi array

        for (int i=0; i< 3 ; i++)
        {
            for (int j=0; j < 3 ; j++)
            {
                System.out.print(arr[i][j] + " ");
            }

            System.out.println();
        }
    }

}
```

Cloning of arrays

When you clone a single-dimensional array, such as Object[], a “deep copy” is performed with the new array containing copies of the original array’s elements as opposed to references.



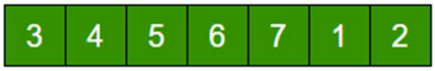



```
package latarray;
//Latihan cloning array
public class latihanar3 {

    public static void main(String[] args) {
        // TODO code application logic here
        int x[] = {1,2,3};
        int xx[][] = {{10,20,30},{40,50}};
        int y[] = x.clone();
        int yy[][] = xx.clone();

        System.out.println(x == y); //perbedaan array
        System.out.println(xx == yy);

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

          for (int i = 0; i < yy.length; i++) {
            for (int j = 0; j < yy[i].length; j++) {
            System.out.print(yy[i][j]+" ");  }
            System.out.println();  }

    }

}
```

Array Rotation

Write a function rotate(ar[], d, n) that rotates arr[] of size n by d elements.

Rotation of the above array by 2 will make array



METHOD 1 (Using temp array)

Input arr[] = [1, 2, 3, 4, 5, 6, 7], d = 2, n =7

1) Store the first d elements in a temp array

temp[] = [1, 2]

2) Shift rest of the arr[]

arr[] = [3, 4, 5, 6, 7, 6, 7]

3) Store back the d elements

arr[] = [3, 4, 5, 6, 7, 1, 2]

METHOD 2 (Rotate one by one)

leftRotate(arr[], d, n)

start

For i = 0 to i < d

Left rotate all elements of arr[] by one

end

To rotate by one, store arr[0] in a temporary variable temp, move arr[1] to arr[0], arr[2] to arr[1] …and finally temp to arr[n-1]
Let us take the same example arr[] = [1, 2, 3, 4, 5, 6, 7], d = 2
Rotate arr[] by one 2 times
We get [2, 3, 4, 5, 6, 7, 1] after first rotation and [3, 4, 5, 6, 7, 1, 2] after second rotation.

```
package latarray;
//Latihan array untuk pergeseran value elemen
//methode rotate one by one
public class latihanar4 {
    public static void geserkekiri(int arr[], int d)
    {
        for (int i = 0; i < d; i++)
            geserkekirisatuan(arr);
    }
```

```
    public static void geserkekirisatuan(int arr[])
    {
        int i, temp;
        int n= arr.length;
        temp = arr[0];
        for (i = 0; i < (n - 1); i++)
            arr[i] = arr[i + 1];
        arr[n-1] = temp;
    }

   public static void printArray(int arr[])
    {
        for (int i = 0; i < arr.length; i++)
            System.out.print(arr[i] + " ");
    }

    public static void main(String[] args) {
        // TODO code application logic here

        int arr[] = { 1, 2, 3, 4, 5, 6, 7 };
        geserkekiri(arr, 2);
        printArray(arr);
    }
    //n array size, d jumlah pergeseran


}
```

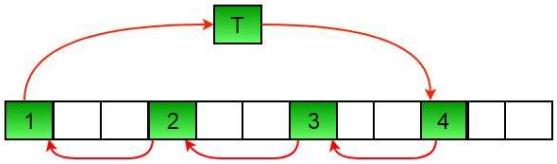
METHOD 3 (A Juggling Algorithm)

This is an extension of method 2. Instead of moving one by one, divide the array in different sets
where number of sets is equal to GCD of n and d and move the elements within sets.
If GCD is 1 as is for the above example array (n = 7 and d =2), then elements will be moved within one set only, we just start with temp = arr[0] and keep moving arr[I+d] to arr[I] and finally store temp at the right place.
Here is an example for n =12 and d = 3. GCD is 3 and

Let arr[] be {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12}

a) Elements are first moved in first set – (See below

diagram for this movement)



arr[] after this step --> {4 2 3 7 5 6 10 8 9 1 11 12}

b) Then in second set.

arr[] after this step --> {4 5 3 7 8 6 10 11 9 1 2 12}

c) Finally in third set.

arr[] after this step --> {4 5 6 7 8 9 10 11 12 1 2 3}

```
package latarray;
//Latihan array untuk pergeseran value elemen
//methode rotate one by one A Juggling Algorithm
public class latihanar5 {
    //d size array, n jumlah yang di geser
    public static void geserkekiri(int arr[], int d)
    {
        int n=arr.length;
        d = d % n;
        int i, j, k, temp;
        int g_c_d = gcd(d, n);
        for (i = 0; i < g_c_d; i++) {
            /* move i-th values of blocks */
            temp = arr[i];
            j = i;
            while (true) {
                k = j + d;
                if (k >= n)
                    k = k - n;
                if (k == i)
                    break;
                arr[j] = arr[k];
                j = k;
            }
            arr[j] = temp;
        }
    }
    public static void geserkekirisatuan(int arr[])
    {
```

```
        int i, temp;
        int n=arr.length;
        temp = arr[0];
        for (i = 0; i < n - 1; i++)
            arr[i] = arr[i + 1];
        arr[n-1] = temp;
    }

    public static void printArray(int arr[])
    {
        for (int i = 0; i < arr.length; i++)
            System.out.print(arr[i] + " ");
    }

    public static int gcd(int d, int n)
    {
        if (n == 0)
            return d;
        else
            return gcd(n, d % n);
    }

     public static void main(String[] args) {
        // TODO code application logic here
        int arr[] = { 1, 2, 3, 4, 5, 6, 7 };
        geserkekiri(arr, 2);
        printArray(arr);
    }
    //n array size, d jumlah pergeseran


}
```

Reversal algorithm for array rotation

Write a function rotate(arr[], d, n) that rotates arr[] of size n by d elements.

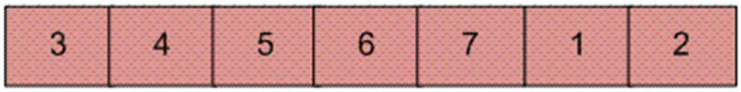
Example :

Input : arr[] = [1, 2, 3, 4, 5, 6, 7]

d = 2

Output : arr[] = [3, 4, 5, 6, 7, 1, 2]

Rotation of the above array by 2 will make array



Method 4 (The Reversal Algorithm) :

Algorithm :

rotate(arr[], d, n)

reverse(arr[], 1, d) ;

reverse(arr[], d + 1, n);

reverse(arr[], 1, n);

Let AB are the two parts of the input array where A = arr[0..d-1] and B = arr[d..n-1]. The idea of the algorithm is :

Reverse A to get ArB, where Ar is reverse of A.

Reverse B to get ArBr, where Br is reverse of B.

Reverse all to get (ArBr) r = BA.

Example :
Let the array be arr[] = [1, 2, 3, 4, 5, 6, 7], d =2 and n = 7
A = [1, 2] and B = [3, 4, 5, 6, 7]

Reverse A, we get ArB = [2, 1, 3, 4, 5, 6, 7]

Reverse B, we get ArBr = [2, 1, 7, 6, 5, 4, 3]

Reverse all, we get (ArBr)r = [3, 4, 5, 6, 7, 1, 2]

Below is the implementation of the above approach :

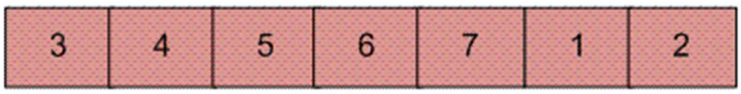
```
package latarray;
//Latihan array untuk pergeseran value elemen
//methode rotate one by one The Reversal Algorithm
public class latihanar6 {
    public static void geserkekiri(int arr[], int d)
    {
        if (d == 0)
            return;
```

```
        int n = arr.length;
        d = d % n;
        memutar(arr, 0, d - 1);
        memutar(arr, d, n - 1);
        memutar(arr, 0, n - 1);
    }
    public static void memutar(int arr[], int start, int end)
    {
       int temp;
        while (start < end) {
            temp = arr[start];
            arr[start] = arr[end];
            arr[end] = temp;
            start++;
            end--;
        }
    }

   public static void printArray(int arr[])
    {
        for (int i = 0; i < arr.length; i++)
            System.out.print(arr[i] + " ");
    }

    public static void main(String[] args) {
        // TODO code application logic here
        int arr[] = { 1, 2, 3, 4, 5, 6, 7 };
        geserkekiri(arr, 2);
        printArray(arr);
    }
    //n array size, d jumlah pergeseran


}
```

Block swap algorithm for array rotation

Write a function rotate(ar[], d, n) that rotates arr[] of size n by d elements.

Rotation of the above array by 2 will make array



Recommended: Please try your approach on ***{IDE}*** first, before moving on to the solution.

Algorithm :
Initialize A = arr[0..d-1] and B = arr[d..n-1]

1) Do following until size of A is equal to size of B

a) If A is shorter, divide B into Bl and Br such that Br is of same length as A. Swap A and Br to change ABlBr into BrBlA. Now A is at its final place, so recur on pieces of B.

b) If A is longer, divide A into Al and Ar such that Al is of same length as B Swap Al and B to change AlArB into BArAl. Now B is at its final place, so recur on pieces of A.

2) Finally when A and B are of equal size, block swap them.

Recursive Implementation:

```
package latarray;
import java.util.*;
//Latihan array untuk pergeseran value elemen
//methode rotate one by one Block swap algorithm
public class latihanar7 {
    public static void geserkekiri(int arr[], int d)
    {
        memutar(arr,0, d, arr.length);
    }
    public static void memutar(int arr[], int i,int d, int n)
    {

        if(d == 0 || d == n)
            return;

        /*If number of elements to be rotated
        is exactly half of array size */
        if(n - d == d)
        {
            swap(arr, i, n - d + i, d);
            return;
        }

        /* If A is shorter*/
```

```
        if(d < n - d)
        {
            swap(arr, i, n - d + i, d);
            memutar(arr, i, d, n - d);
        }
        else /* If B is shorter*/
        {
            swap(arr, i, d, n - d);
            memutar(arr, n - d + i, 2 * d - n, d); /*This is tricky*/
        }
    }

    public static void printArray(int arr[])
    {
        for (int i = 0; i < arr.length; i++)
            System.out.print(arr[i] + " ");
    }

    public static void swap(int arr[], int fi,
                            int si, int d)
   {
     int i, temp;
     for(i = 0; i < d; i++)
     {
        temp = arr[fi + i];
        arr[fi + i] = arr[si + i];
        arr[si + i] = temp;
     }
  }

    public static void main(String[] args) {
        // TODO code application logic here
        int arr[] = { 1, 2, 3, 4, 5, 6, 7 };
        geserkekiri(arr,2);
        printArray(arr);
    }
    //n array size, d jumlah pergeseran


}
```