```
class ArrayRotation{

	public void rotate(int arr[], int d){

		int len = arr.length;
		reverse(arr,0,d-1);
		reverse(arr,d,len-1);
		reverse(arr,0,len-1);
	}

	public void reverse(int arr[], int start, int end){
		int temp;
		for(int i=start,j=end;i<=j;i++,j--){
			temp = arr[i];
			arr[i] = arr[j];
			arr[j] = temp;
		}
	}

	public int searchInRotated(int arr[], int low, int high, int item){

		if(high < low)
			return -1;
		int mid = (low + high)/2;
		if(arr[mid] == item)
			return mid;
		if(arr[low] < arr[mid]){
			if(arr[low] <= item && item <= arr[mid])
				return searchInRotated(arr,low,mid-1,item);
			else
				return searchInRotated(arr,mid+1,high,item);
		}
		else{
			if(arr[mid] <= item && item <= arr[high])
				return searchInRotated(arr,mid+1,high,item);
			else
				return searchInRotated(arr,low,mid-1,item);
		}
	}

	public int countSumPairs(int arr[],int sum){

		int len = arr.length;
		int i;
```

```
        for(i=0;i<len;i++){
                if(arr[i]>arr[i+1])
                        break;
        }

        int low = (i+1)%len;
        int high = i;
        int count = 0;

        while(low != high){
                if(arr[low] + arr[high] == sum){
                        count++;

                        if(low == (high+len-1)%len)
                                return count;

                        low = (low+1)%len;
                        high = (high+len-1)%len;
                }

                if(arr[low] + arr[high] < sum)
                        low = (low+1)%len;
                else
                        high = (high+len-1)%len;
        }

        return count;
}

int countRotations(int arr[], int low, int high){
        if(high < low)
                return 0;
        if(high == low)
                return low;

        int mid = (low + high)/2;

        if(arr[mid] > arr[mid-1] && arr[mid] > arr[mid+1])
                return mid+1;
        if(arr[mid] < arr[mid-1] && arr[mid] < arr[mid+1])
                return mid;
        if (arr[high] > arr[mid]) {
                return countRotations(arr,low,mid-1);
        }
```

```
        else{
                return countRotations(arr,mid+1,high);
        }
}

int maximumSum(int arr[]){

        int len = arr.length;
        int arrSum = 0;
        int currVal = 0;
        int maxVal = 0;

        for(int i=0;i<len;i++){
                arrSum = arrSum + arr[i];
                currVal = currVal + (i*arr[i]);
        }

        maxVal = currVal;

        for(int j=1; j<len; j++){
                currVal = currVal + arrSum - (len*arr[len-j]);
                if(currVal > maxVal)
                        maxVal = currVal;
        }

        return maxVal;
}

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

public static void main(String args[]){

        ArrayRotation obj = new ArrayRotation();
        int arr[] = {1,2,3,4,5,6,7,8};
        obj.rotate(arr,3);
        obj.printArray(arr);
        int loc = obj.searchInRotated(arr,0,arr.length-1,6);
        System.out.println("Element found at: " + loc + " index");
        int count = obj.countSumPairs(arr,10);
```

```
		System.out.println("Number of pairs with given sum: " + count);
		int rotations = obj.countRotations(arr,0,arr.length-1);
		System.out.println("Number of rotations: " + rotations);
		int maxSum = obj.maximumSum(arr);
		System.out.println("Maximum Sum of type i*arr[i]: " + maxSum);

	}
}



import java.util.*;

class ArrayArrangement {

	public void alternatePosNeg(int arr[]){

		int len = arr.length;
		int i = -1; int temp = 0;

		for(int j=0; j<len; j++){
			if(arr[j] < 0){
				i++;
				temp = arr[i];
				arr[i] = arr[j];
				arr[j] = temp;
			}
		}

		int pos = i+1; int neg = 0;
		while(pos < len && neg < pos && arr[neg] < 0){
			temp = arr[neg];
			arr[neg] = arr[pos];
			arr[pos] = temp;
			pos += 1;
			neg += 2;
		}
	}

	public void moveZeroesToEnd(int arr[]){
		int temp = 0;
		int count = 0;
		for(int i=0; i<arr.length; i++){
			if(arr[i] != 0){
```

```
                        temp = arr[i];
                        arr[i] = arr[count];
                        arr[count] = temp;
                        count += 1;
                    }
                }
            }


    public void indexArrangement(int arr[], int index[]){

            for(int i=0; i<arr.length; i++){

                    while(index[i] != i){

                        int  oldIndex  = index[index[i]];
      char oldValue  = (char)arr[index[i]];

                        index[index[i]] = index[i];
                        arr[index[i]] = arr[i];

                        index[i] = oldIndex;
                        arr[i] = oldValue;
                    }
            }
    }

    public void formLargestNumber(Vector<String> numbers){

            Collections.sort(numbers, new Comparator<String>() {
   @Override
   public int compare(String X, String Y) {
      String XY = X + Y;
      String YX = Y + X;

      return XY.compareTo(YX) > 0 ? -1 : 1;
   }
});

Iterator<String> iter = numbers.iterator();
while(iter.hasNext()){
   System.out.print(iter.next());
}
System.out.println();
```

```
	}

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

	public static void main(String args[]){
		ArrayArrangement obj = new ArrayArrangement();
		int arr[] = {-7,1,3,4,-5,-2,9,-1};
		obj.alternatePosNeg(arr);
		obj.printArray(arr);
		int arr2[] = {0,1,2,3,4,0,5,0,7,0};
		obj.moveZeroesToEnd(arr2);
		obj.printArray(arr2);
		int arr3[] = {10,20,30,40,50,60};
		int index[] = {4,2,0,5,1,3};
		obj.indexArrangement(arr3,index);
		obj.printArray(arr3);
		Vector<String> vector = new Vector<String>();
		vector.add("12");
		vector.add("78");
		vector.add("342");
		vector.add("569");
		obj.formLargestNumber(vector);
	}
}


class ArrayOrderStatistics {

	public int kthSmallest(int arr[],int low,int high,int k){

		if(k > 0 && k <= high-low+1){

			int pos = partition(arr,low,high);

			if(pos-low == k-1)
				return arr[pos];

			if(pos - low > k-1)
				return kthSmallest(arr,low,pos-1,k);
```

```
                return kthSmallest(arr,pos+1,high,k-pos+low-1);
            }

            return Integer.MAX_VALUE;
        }

        public int partition(int arr[], int low, int high){

            int x = arr[high];
            int i = low;
            int temp = 0;
            for(int j=low;j<high;j++){
                if(arr[j] <= x){
                    temp = arr[j];
                    arr[j] = arr[i];
                    arr[i] = temp;
                    i++;
                }
            }

            temp = arr[i];
            arr[i] = arr[high];
            arr[high] = temp;

            return i;
        }

        public static void main(String args[]){
            ArrayOrderStatistics obj = new ArrayOrderStatistics();
            int arr[] = {3,1,8,5,0,20,11,45};
            int kth = obj.kthSmallest(arr,0,arr.length-1,6);
            System.out.println("6th smallest: " + kth);
        }
}
```

```
import java.util.*;

class ArrayQuestions {

	public void checkPairWithGivenSum(int arr[], int sum){

		int temp;
		HashSet<Integer> set = new HashSet<Integer>();

		for(int i=0; i<arr.length; i++){

			temp = sum - arr[i];
			if(temp >= 0 && set.contains(temp)){
				System.out.println("Pair with given sum: " + arr[i] + " and " +
temp);
			}

			set.add(arr[i]);

		}
	}

	public int findPosition(int arr[], int key){

		int low = 0, high = 1;
		int value = arr[low];

		while(key > value){
			low = high;
			high = 2*high;
			value = arr[high];
		}

		return binarySearch(arr,low,high,key);
	}

	public int binarySearch(int arr[], int low, int high,int key){

		if(high>=low){
			int mid = (low+high)/2;
			if(arr[mid] == key)
				return mid;
			if(arr[mid] > key)
				return binarySearch(arr,low,mid-1,key);
```

```
                else
                        return binarySearch(arr,mid+1,high,key);
        }
        return -1;
}

public int equilibriumMaxSum(int arr[]){

        int sum = 0;
        int preSum = 0;
        int result = Integer.MIN_VALUE;
        for(int i=0; i<arr.length; i++)
                sum += arr[i];

        for(int i=0; i<arr.length; i++){
                preSum += arr[i];

                if(preSum == sum)
                        result = Math.max(result,preSum);

                sum -= arr[i];
        }

        return result;
}

public int equilibriumIndex(int arr[]){
        int sum = 0, leftSum = 0;
        for(int i=0; i<arr.length; i++)
                sum += arr[i];

        for(int i=0; i<arr.length; i++){
                sum -= arr[i];

                if(leftSum == sum)
                        return i;

                leftSum += arr[i];

        }

        return -1;
}
```

```
public void leaders(int arr[]){
	int size = arr.length;
	int max_from_right = arr[size-1];
	System.out.println("Leaders: ");
	System.out.print(max_from_right + " ");

	for(int i=size-2; i>=0; i--){
		if(arr[i] > max_from_right){
			max_from_right = arr[i];
			System.out.print(max_from_right + " ");
		}
	}

	System.out.println();
}

public void printMajority(int arr[]){
	int size = arr.length;
	int candidate = findMajority(arr,size);
	if(isMajority(arr,size,candidate))
		System.out.println("Majority Element: " + candidate);
	else
		System.out.println("No majority element");
}

public int findMajority(int arr[], int size){
	int majority_index = 0, count = 1;
	for(int i=1; i<size; i++){
		if(arr[majority_index] == arr[i])
			count++;
		else
			count--;

		if(count == 0){
			majority_index = i;
			count = 1;
		}
	}

	return arr[majority_index];
}

public boolean isMajority(int arr[], int size, int candidate){
	int count = 0;
```

```
		for(int i=0; i<size; i++){
			if(arr[i] == candidate)
				count++;
		}

		if(count > size/2)
			return true;
		else
			return false;
	}

	public int peak(int arr[], int low, int high, int size){
		int mid = low + (high-low)/2;

		if((mid == 0 || arr[mid-1] <= arr[mid]) && (mid == size-1 || arr[mid+1] <=
arr[mid]))
			return mid;

		if(mid > 0 && arr[mid-1] > arr[mid])
			return peak(arr,low,mid-1,size);
		else
			return peak(arr,mid+1,high,size);
	}

	public static void main(String args[]){
		ArrayQuestions obj = new ArrayQuestions();
		int arr[] = {12,2,11,8,7,4,9};
		obj.checkPairWithGivenSum(arr,10);
		int arr2[] = {3, 5, 7, 9, 10, 90,100, 130, 140, 160, 170};
		int pos = obj.findPosition(arr2,10);
		System.out.println("Position: " + pos);
		int arr3[] = { -2, 5, 3, 1,2, 6, -4, 2 };
		System.out.println("Equilibrium Sum: " + obj.equilibriumMaxSum(arr3));
		System.out.println("Equilibrium index: " + obj.equilibriumIndex(arr3));
		obj.leaders(arr);
		int arr4[] = {1,2,2,2,2,4,5,6,2,3,2,7,2,3,2};
		obj.printMajority(arr4);
		int peakIndex = obj.peak(arr,0,arr.length-1,arr.length);
		System.out.println("Peak Value: " + arr[peakIndex]);

	}

}
```

```
class MatrixQuestions {

	public void rotate90Anticlockwise(int mat[][]){
		transpose(mat);
		reverseColumns(mat);
	}

	public void rotate90Clockwise(int mat[][]){
		transpose(mat);
		reverseRows(mat);
	}

	public void rotate180(int mat[][]){
		transpose(mat);
		reverseColumns(mat);
		transpose(mat);
		reverseColumns(mat);
	}

	public void printMatrix(int mat[][]){

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

	public void transpose(int mat[][]){
		for(int i=0;i<mat.length;i++){
			for(int j=i;j<mat[0].length;j++){
				int temp = mat[j][i];
				mat[j][i] = mat[i][j];
				mat[i][j] = temp;
			}
		}
	}

	public void reverseColumns(int mat[][]){

		for(int i=0;i<mat[0].length;i++){
			for(int j=0,k=mat[0].length-1;j<k;j++,k--){
				int temp = mat[j][i];
```

```
                    mat[j][i] = mat[k][i];
                    mat[k][i] = temp;
                }
        }
}

public void reverseRows(int mat[][]){

        for(int i=0;i<mat.length;i++){
                for(int j=0,k=mat.length-1;j<k;j++,k--){
                    int temp = mat[i][j];
                    mat[i][j] = mat[i][k];
                    mat[i][k] = temp;
                }
        }
}

public void printSpiral(int matrix[][]){
        int rowStart = 0, colStart = 0;
        int rowLength = matrix.length-1, colLength = matrix[0].length-1;
        while(rowStart <= rowLength && colStart <= colLength){
                for(int i=rowStart;i<=colLength;i++)
                    System.out.print(matrix[rowStart][i] + " ");
                for(int j=rowStart+1;j<=rowLength;j++)
                    System.out.print(matrix[j][colLength] + " ");
                if(rowStart+1<=rowLength){
                    for(int k=colLength-1;k>=colStart;k--)
                        System.out.print(matrix[rowLength][k] + " ");
                }
                if(colStart+1<=colLength){
                    for(int l=rowLength-1;l>rowStart;l--)
                        System.out.print(matrix[l][colStart] + " ");
                }
                rowStart++;
                rowLength--;
                colStart++;
                colLength--;
        }
}

public static void main(String args[]){
        int matrix [][] = { {1,  2,  3,  4,  5},
                                                {6,  7,  8,  9,  10},
                                                {11, 12, 13, 14, 15},
```

```
                                                {16, 17, 18, 19, 20},
                                                {21, 22, 23, 24, 25}};
                MatrixQuestions obj = new MatrixQuestions();
                //obj.rotate90Clockwise(matrix);
                //obj.printMatrix(matrix);
                //obj.rotate90Anticlockwise(matrix);
                //obj.printMatrix(matrix);
                //obj.rotate180(matrix);
                //obj.printMatrix(matrix);
                obj.printSpiral(matrix);
        }

}
```