Utils

- Stack Algo

```java
public class Stack {
    private static final int ERROR_CODE = -111111;
    private int size;
    private int top;
    private int[] arr;

    Stack(int size) {
        this.size = size;
        arr = new int[size];
        top = -1;
    }

    public int getSize() {
        return size;
    }

    public int getTop() {
        return top;
    }

    public boolean isFull() {
        return top == size - 1;
    }

    public boolean isEmpty() {
        return top == -1;
    }

    public void push(int val) {
        if (!isFull()) {
            arr[++top] = val;
        } else {
            System.out.println("\nStackOverflowed");
        }
    }

    public int pop() {
        if (!isEmpty()) {
            return arr[top--];
        } else {
            System.out.println("\nStackUnderflow");
            return ERROR_CODE;
        }
    }

    public int peek(int pos) {
        if (!isEmpty()) {
            return arr[top - pos + 1];
        }
        return ERROR_CODE;
    }
}
```

Q1. Write a recursive function that takes as input a queue and rearranges it so that it is in reverse order. Hint: deque() the first element, recursively reverse the queue, and the enqueuer the first element.

```
public class q1 {
    public static void revQueue(Queue q) {
        if (q.isEmpty())
            return;
        int element = q.deQueue();
        revQueue(q);
        q.enQueue(element);
    }

    public static void main(String[] args) {
        int size = 5;
        Queue q = new Queue(size);
        q.enQueue(3);
        q.enQueue(5);
        q.enQueue(8);
        q.enQueue(2);
        q.enQueue(7);

        revQueue(q);
        q.display();
    }
}
```

Q2. Write a program from scratch that reverses the words in a sentence

```
public static void charReverse(String string) {
    String[] array = string.split(" ");

    for (int i = 0; i < array.length; i++) {
        char[] word = array[i].toCharArray();
        for (int j = word.length - 1; j >= 0; j--) {
            System.out.print(word[j]);
        }
        System.out.print(" ");
    }
}
```

Q4. Write the function transforming a decimal number into a binary number by using stack

```
public static void binStack(int num){
    Stack stack = new Stack(20);
    while (num > 0) {
        int r = num % 2;
        stack.push(r);
        num = num / 2;
    }

    while (!(stack.isEmpty())) {
        System.out.print(stack.pop());
    }
}
```

Q5. Write the function that removes all even numbers from the given stack. The mutual order of odd numbers must stay unchanged. The function returns the number of removed numbers.

```
public static void main(String[] args) {
    Stack stack = new Stack(10);
    Stack temp = new Stack(10);
    int counter = 0;

    stack.push(1);
    stack.push(2);
    stack.push(3);
    stack.push(4);
    stack.push(5);

    while (!stack.isEmpty()) {
        int val = stack.pop();
        if (val % 2 != 0) {
            temp.push(val);
        } else {
            counter++;
        }
    }

    while (!temp.isEmpty()) {
        stack.push(temp.pop());
    }

    System.out.println("After Removing Even Integers");
    while (!stack.isEmpty()) {
        System.out.print(stack.pop() + " ");
    }

    System.out.println("\nTotal Even Integers Removed: " + counter);
}
```

Q6. Write the function that returns duplicate stack of the given stack. Duplicate stack contains the same elements as the original stack, and in the same order. The original stack must stay unchanged.

```
public class q6 {

    private static Stack copyStack(Stack stack) {
        int size = stack.getSize();
        Stack copy_stack = new Stack(size);

        for (int i = stack.getTop() + 1; i > 0; i--) {
            copy_stack.push(stack.peek(i));
        }
        return copy_stack;
    }

    public static void main(String[] args) {
        Stack stack = new Stack(10);

        stack.push(1);
        stack.push(2);
        stack.push(3);
        stack.push(4);
        stack.push(5);

        Stack stackCopy = copyStack(stack);

        System.out.println("Original Stack");
        System.out.println("Size: " + stack.getSize());
        System.out.println("Top: " + stack.getTop());
        System.out.print("Elements: ");

        while (!stack.isEmpty()) {
            System.out.print(stack.pop() + " ");
        }

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

        System.out.println("Duplicate Stack");
        System.out.println("Size: " + stackCopy.getSize());
        System.out.println("Top: " + stackCopy.getTop());
        System.out.print("Elements: ");

        while (!stackCopy.isEmpty()) {
            System.out.print(stackCopy.pop() + " ");
        }
    }

}
```

Q8. A palindrome is a phrase that reads the same forward and backward (examples: ‘racecar’, ‘radar’, ‘noon’, or ‘rats live on no evil star’). By extension we call every string a palindrome that reads the same from left to right and from right to left. Develop a recursive algorithm that takes as input a string and decides whether the string is a palindrome. Write down your algorithm in pseudocode.

```
public class q8 {
    public static boolean findPalindrome(String word) {
        int length = word.length();

        if (length < 2) {
            return true;
        }

        if (word.charAt(0) != word.charAt(length - 1)) {
            return false;
        }

        return findpl(word.substring(1, length - 1));
    }

    public static void main(String[] args) {
        String word = "radar";
        System.out.println(findPalindrome(word));
    }
}
```

Q9. Write a program to remove the duplicate elements of a given array and return the new length of the array. Sample array: [20, 20, 30, 40, 50, 50, 50] After removing the duplicate elements, the program should return 4 as the new length of the array.

```
import java.util.Arrays;

public static void set(int arr) {
    Arrays.sort(arr);
    int n = arr.length;
    int count = 0;

    if (n < 2) {
        return n;
    }

    for (int i = 0; i < n - 1; i++) {
        if (arr[i] != arr[i + 1]) {
            count++;
        }
    }
    count++;
    return count;
}
```

Q10. Write a program for Matrix multiplication of two matrices having different sizes?

```
public static void main(int[][] mat1, int[][] mat2) {
    if (mat1[0].length != mat2.length) {
        return -1;
    }

    for (int i = 0; i < mat1.length; i++) {
        for (int j = 0; j < mat2[i].length; j++) {
            int sum = 0;
            for (int k = 0; k < mat2.length; k++) {
                sum = sum + mat1[i][k] * mat2[k][j];
            }
            System.out.print(sum + " ");
        }
        System.out.println();
    }
}
```