Utils

- Stack Algo

```
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;
    }
}
```

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

```
public static void wordRev(String str) {
    String[] strArr = str.split(" ");

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

Q3. Use your stack implementation to implement a queue using the Stack.

```
public class q3 {
    public static void enQueue(Stack stack, int x) {
    stack.push(x);
    }

    public static int deQueue(Stack stack) {
        Stack temp = new Stack(stack.size());
        while (!stack.isEmpty()) {
            temp.push(stack.pop());
        }
        int x = temp.pop();
        while (!temp.isEmpty()) {
            stack.push(temp.pop());
        }
        return x;
    }

    public static void main(String[] args) {
        int size = 5;
        Stack stack = new Stack(size);
        enQueue(stack, 5);
        enQueue(stack, 3);
        enQueue(stack, 7);
        deQueue(stack);
        deQueue(stack);

        stack.printStack();
```

```
    }
}
```

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.

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

    stack.push(1);
    stack.push(2);
    stack.push(3);
    stack.push(4);
    stack.push(5);
    stack.push(6);
    stack.push(7);
    stack.push(8);
    stack.push(9);
    stack.push(10);

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

```
        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: " + removeCounter);
    }
```

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 duplicateStack(Stack stack) {
        int size = stack.getSize();
        Stack dulStack = new Stack(size);

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

    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 dulStack = duplicateStack(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: " + dulStack.getSize());
```

```
        System.out.println("Top: " + dulStack.getTop());
        System.out.print("Elements: ");

        while (!dulStack.isEmpty()) {
            System.out.print(dulStack.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 findpl(String pl) {
        int len = pl.length();

        if (len < 2) {
            return true;
        }

        if (pl.charAt(0) != pl.charAt(len - 1)) {
            return false;
        }

        return findpl(pl.substring(1, len - 1));
    }

    public static void main(String[] args) {
        String pl = "noon";
        System.out.println(findpl(pl));
    }
}
```

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) {
        System.out.println("Impossible For Computers");
        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();
    }
}
```