Set-1

1.Write a recursive method to check if a given string is a palindrome (reads the same forward and backward).

Test Cases:

- "racecar" → true
- "hello" → false

```java
public class PalindromeChecker {

    public static boolean isPalindrome(String s) {
        if (s.length() <= 1) {
            return true;
        }
        if (s.charAt(0) != s.charAt(s.length() - 1)) {
            return false;
        }
        return isPalindrome(s.substring(1, s.length() - 1));
    }

    public static void main(String[] args) {
        System.out.println(isPalindrome("racecar"));
        System.out.println(isPalindrome("hello"));
    }
}

```

```
java -cp /tmp/ij33El4Mbf/PalindromeChe
true
false

=== Code Execution Successful ===
```

2.Write a method to validate an email address using regular expressions.

Test Cases:

- "test@example.com" → true
- "invalid-email" → false

```java
import java.util.regex.Pattern;

public class EmailValidator {

    public static boolean isValidEmail(String email) {
        String emailRegex = "^[A-Za-z0-9+_.-]+@[A-Za-z0-9.-]+$";
        Pattern pattern = Pattern.compile(emailRegex);
        return pattern.matcher(email).matches();
    }

    public static void main(String[] args) {
        System.out.println(isValidEmail("test@example.com"));
        System.out.println(isValidEmail("invalid-email"));
    }
}

```

```
java -cp /tmp/AjIWhjBNbn/EmailValidator
true
false

=== Code Execution Successful ===
```

3. Write a recursive method to calculate the factorial of a given number.

Test Cases:

- **5 → 120**
- **3 → 6**

```java
public class FactorialCalculator {

    public static int factorial(int n) {
        if (n == 0 || n == 1) {
            return 1;
        }
        return n * factorial(n - 1);
    }

    public static void main(String[] args) {
        System.out.println(factorial(5));
        System.out.println(factorial(3));
    }
}

```

```
java -cp /tmp/wWOruI06Ne/FactorialCalculator
120
6

=== Code Execution Successful ===
```

4. Write a method to compress a string using the counts of repeated characters. For example, "aabcccccaaa" should become "a2b1c5a3".

Test Cases:

- **"aabcccccaaa" → "a2b1c5a3"**
- **"abcd" → "abcd"**

```java
public class StringCompressor {

    public static String compressString(String s) {
        StringBuilder compressed = new StringBuilder();
        int countConsecutive = 0;

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

            if (i + 1 >= s.length() || s.charAt(i) != s.charAt(i + 1)) {
                compressed.append(s.charAt(i));
                compressed.append(countConsecutive);
                countConsecutive = 0;
            }
        }

        return compressed.length() < s.length() ? compressed.toString() : s;
    }

    public static void main(String[] args) {
        System.out.println(compressString("aabcccccaaa"));
        System.out.println(compressString("abcd"));
    }
}

```

```
java -cp /tmp/wfZNzTSdTD/StringCompressor
a2b1c5a3
abcd

=== Code Execution Successful ===
```

5.Write a method to check if a given string matches a specified pattern using regular expressions. For example, check if the string is a valid phone number.

Test Cases:

- **"123-456-7890" → true**
- **"1234567890" → false**

```java
import java.util.regex.Pattern;

public class PhoneNumberValidator {

    public static boolean isValidPhoneNumber(String phoneNumber) {
        String phoneRegex = "^\\d{3}-\\d{3}-\\d{4}$";
        Pattern pattern = Pattern.compile(phoneRegex);
        return pattern.matcher(phoneNumber).matches();
    }

    public static void main(String[] args) {
        System.out.println(isValidPhoneNumber("123-456-7890"));
        System.out.println(isValidPhoneNumber("1234567890"));
    }
}
```

```
java -cp /tmp/gNgkbwtcQP/PhoneNumberValidator
true
false

=== Code Execution Successful ===
```