Class Test - 9

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

PROGRAM:

```
public class PalindromeChecker {

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

    public static boolean isPalindrome(String str) {
        return isPalindromeHelper(str, 0, str.length() - 1);
    }

    private static boolean isPalindromeHelper(String str, int left,
int right) {
        if (left >= right) {
            return true;
        }
        if (str.charAt(left) != str.charAt(right)) {
            return false;
        }
        return isPalindromeHelper(str, left + 1, right - 1);
    }
}
```

OUTPUT:

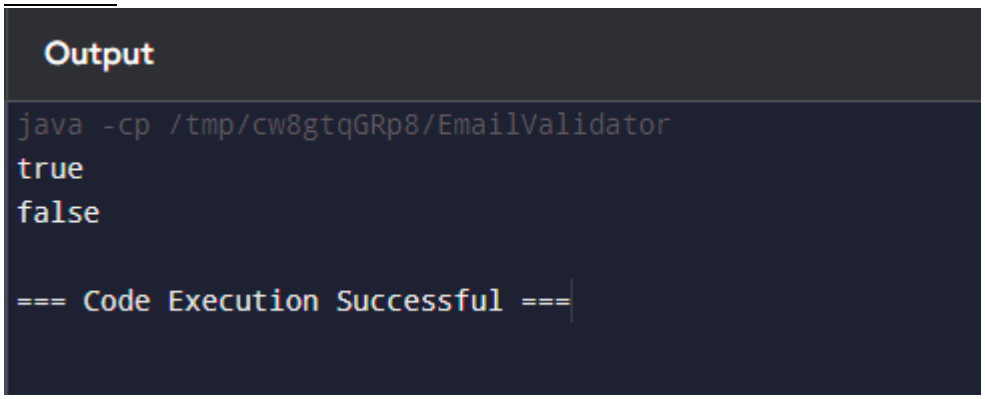
```
Output

java -cp /tmp/3qXfewJalv/PalindromeChecker
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

PROGRAM:

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

public class EmailValidator {

    public static boolean isValidEmail(String email) {
        String emailRegex = "^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\\.[a-zA-Z]{2,}$";
        Pattern pattern = Pattern.compile(emailRegex);
        return pattern.matcher(email).matches();
    }

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

OUTPUT:



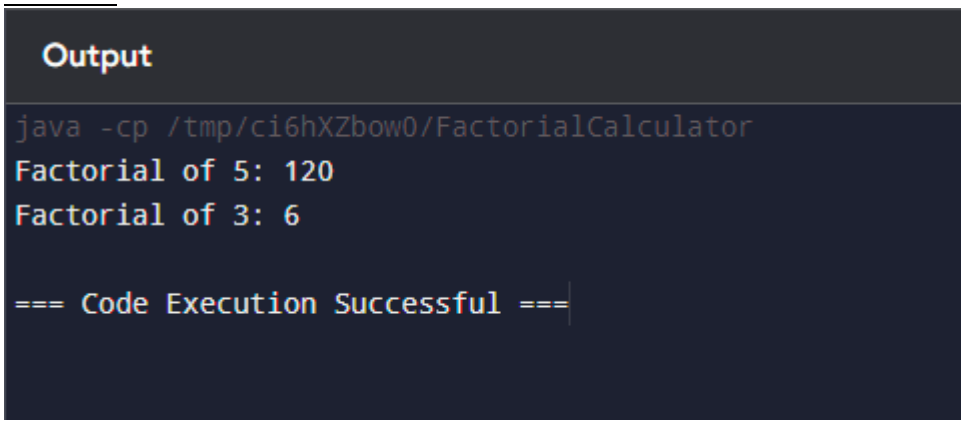
3. Write a recursive method to calculate the factorial of a given number.
Test Cases:
- 5 → 120

● 3 → 6

PROGRAM:

```
public class FactorialCalculator {

    public static void main(String[] args) {
        System.out.println("Factorial of 5: " + factorial(5)); //
120
        System.out.println("Factorial of 3: " + factorial(3)); // 6
    }

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

OUTPUT:



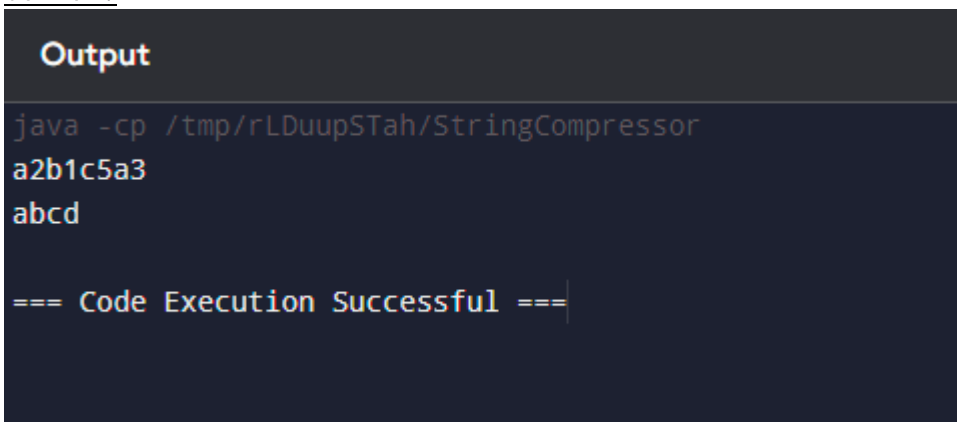
4. Write a method to compress a string using the counts of repeated characters. For example, "aabccccaaa" should become "a2b1c5a3".
Test Cases:
● "aabccccaaa" → "a2b1c5a3"
● "abcd" → "abcd"

PROGRAM:

```
public class StringCompressor {

    public static String compressString(String input) {
        if (input == null || input.isEmpty()) {
```

```
            return input;
        }

        StringBuilder compressed = new StringBuilder();
        int count = 1;

        for (int i = 1; i < input.length(); i++) {
            if (input.charAt(i) == input.charAt(i - 1)) {
                count++;
            } else {
                compressed.append(input.charAt(i  - 1)).append(count);
                count = 1;
            }
        }
        compressed.append(input.charAt(input.length() -
1)).append(count);

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

    public static void main(String[] args) {
        System.out.println(compressString("aabccccccaaa")); // Output:
a2b1c5a3
        System.out.println(compressString("abcd")); // Output: abcd
    }
}
```

OUTPUT:



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

PROGRAM:

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

public class StringPatternChecker {

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

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

OUTPUT: