**SIMATS ENGINEERING SAVEETHA INSTITUTE OF MEDICAL AND TECHNICAL SCIENCES CHENNAI-602105**

**CSA0961 Programming in java for Distributed Applications**

**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**

**Code:**

public class PalindromeChecker {

public static boolean isPalindrome(String s) {

return isPalindromeHelper(s, 0, s.length() - 1);

}

private static boolean isPalindromeHelper(String s, int start, int end) {

if (start >= end) {

return true; // Base case: if pointers cross, it's a palindrome

}

if (s.charAt(start) != s.charAt(end)) {

return false; // Characters don't match

}

return isPalindromeHelper(s, start + 1, end - 1); // Recursive call

}

public static void main(String[] args) {

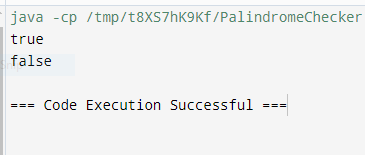
System.out.println(isPalindrome("racecar")); // true

System.out.println(isPalindrome("hello")); // false

}

}

**Output:**



**2. Write a method to validate an email address using regular expressions. Test Cases: "test@example.com" → true "invalid-email" → false Medium**

**Code:**

import java.util.regex.Pattern;

import java.util.regex.Matcher;

public class EmailValidator {

private static final String EMAIL\_REGEX = "^[a-zA-Z0-9.\_%+-]+@[a-zA-Z0-9.-]+\\.[a-zA-Z]{2,}$";

public static boolean isValidEmail(String email) {

Pattern pattern = Pattern.compile(EMAIL\_REGEX);

Matcher matcher = pattern.matcher(email);

return matcher.matches();

}

public static void main(String[] args) {

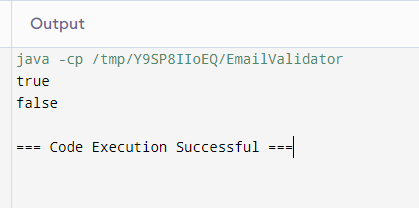
System.out.println(isValidEmail("test@example.com")); // true

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 Easy**

**Code:**

public class FactorialCalculator {

public static int factorial(int n) {

if (n == 0) {

return 1;

}

return n \* factorial(n - 1);

}

public static void main(String[] args) {

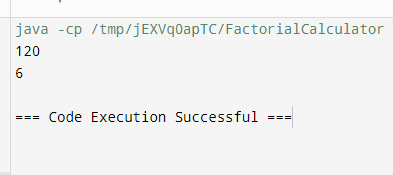
System.out.println(factorial(5));

System.out.println(factorial(3));

}

}

**Output:**

****

**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" Medium**

**Code:**

public class StringCompressor {

public static String compress(String s) {

StringBuilder compressed = new StringBuilder();

int count = 1;

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

if (i + 1 < s.length() && s.charAt(i) == s.charAt(i + 1)) {

count++;

} else {

compressed.append(s.charAt(i)).append(count);

count = 1;

}

}

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

}

public static void main(String[] args) {

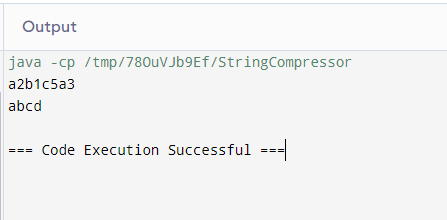
System.out.println(compress("aabcccccaaa"));

System.out.println(compress("abcd"));

}

}

**Output:**

****

**5.5 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**

**Code:**

import java.util.regex.Pattern;

import java.util.regex.Matcher;

public class PhoneNumberValidator {

private static final String PHONE\_NUMBER\_REGEX = "^\\d{3}-\\d{3}-\\d{4}$";

public static boolean isValidPhoneNumber(String phoneNumber) {

Pattern pattern = Pattern.compile(PHONE\_NUMBER\_REGEX);

Matcher matcher = pattern.matcher(phoneNumber);

return matcher.matches();

}

public static void main(String[] args) {

System.out.println(isValidPhoneNumber("123-456-7890"));

System.out.println(isValidPhoneNumber("1234567890"));

}

}

**Output:**

