**Strings - Handling strings and common string functions**

**Functions - parameter passing, return**

**1. Count Vowels and Consonants**

**Problem:**

**Write a Java program to count the number of vowels and consonants in a given string.**

|  |
| --- |
| **public class Constants {  public static void main(String[] args) {  String input = "Example String";  int vowels = 0, consonants = 0;   input = input.toLowerCase();  for (char c : input.toCharArray()) {  if (c >= 'a' && c <= 'z') {  if (c == 'a' || c == 'e' || c == 'i' || c == 'o' || c == 'u') {  vowels++;  } else {  consonants++;  }  }  }   System.out.println("Vowels: " + vowels);  System.out.println("Consonants: " + consonants);  } }** |

**2. Reverse a String**

**Problem:**

**Write a Java program to reverse a given string without using any built-in reverse functions.**

|  |
| --- |
| **public class Rverse {  public static void main(String[] args) {  String input = "example";  String reversed = reverseString(input);  System.out.println("Reversed String: " + reversed);  }   public static String reverseString(String str) {  char[] chars = str.toCharArray();  StringBuilder reversed = new StringBuilder();  for (int i = chars.length - 1; i >= 0; i--) {  reversed.append(chars[i]);  }  return reversed.toString();  } }** |

**3. Palindrome String Check**

**Problem:**

**Write a Java program to check if a given string is a palindrome (a string that reads the same forward and backward).**

|  |
| --- |
| **import java.util.Scanner;   public class Palindrome {  public static void main(String[] args) {  Scanner scanner = new Scanner(System.in);  System.out.print("Enter a string: ");  String input = scanner.nextLine();  String reversed = new StringBuilder(input).reverse().toString();  if (input.equals(reversed)) {  System.out.println("The string is a palindrome.");  } else {  System.out.println("The string is not a palindrome.");  }  scanner.close();  } }** |

**4. Remove Duplicates from a String**

**Problem:**

**Write a Java program to remove all duplicate characters from a given string and return the modified string.**

|  |
| --- |
| **public class Duplicates {  public static String removeDuplicates(String input) {  StringBuilder result = new StringBuilder();  boolean[] seen = new boolean[256];  for (char c : input.toCharArray()) {  if (!seen[c]) {  seen[c] = true;  result.append(c);  }  }  return result.toString();  }   public static void main(String[] args) {  String input = "programming";  System.out.println(removeDuplicates(input));  } }** |

**5. Find the Longest Word in a Sentence**

**Problem:**

**Write a Java program that takes a sentence as input and returns the longest word in the sentence.**

|  |
| --- |
| **import java.util.Scanner;  public class Long {  public static void main(String[] args) {  Scanner scanner = new Scanner(System.in);  System.out.println("Enter a sentence:");  String sentence = scanner.nextLine();  String[] words = sentence.split(" ");  String longestWord = "";   for (String word : words) {  if (word.length() > longestWord.length()) {  longestWord = word;  }  }   System.out.println("The longest word is: " + longestWord);  } }** |

**6. Find Substring Occurrences**

**Problem:**

**Write a Java program to count how many times a given substring occurs in a string.**

|  |
| --- |
| **public class SubString {  public static void main(String[] args) {  String str = "hellohellohello";  String subStr = "hello";  int count = 0;  int index = 0;   while ((index = str.indexOf(subStr, index)) != -1) {  count++;  index += subStr.length();  }   System.out.println("Occurrences: " + count);  } }** |

**7. Toggle Case of Characters**

**Problem:**

**Write a Java program to toggle the case of each character in a given string. Convert uppercase letters to lowercase and vice versa.**

|  |
| --- |
| **import java.util.Scanner;    public class Toggle\_case {  public static void main(String[] args) {  Scanner scanner = new Scanner(System.in);  System.out.print("Enter a string: ");  String input = scanner.nextLine();  StringBuilder toggledString = new StringBuilder();   for (char c : input.toCharArray()) {  if (Character.isUpperCase(c)) {  toggledString.append(Character.toLowerCase(c));  } else if (Character.isLowerCase(c)) {  toggledString.append(Character.toUpperCase(c));  } else {  toggledString.append(c);  }  }   System.out.println("Toggled case string: " + toggledString);  scanner.close();  } }** |

**8. Compare Two Strings**

**Problem:**

**Write a Java program to compare two strings lexicographically (dictionary order) without using built-in compare methods.**

**Example Input:**

**String 1: "apple"**

**String 2: "banana"**

**Expected Output:**

**"apple" comes before "banana" in lexicographical order**

|  |
| --- |
| **public class Order {  public static void main(String[] args) {  String str1 = "apple";  String str2 = "banana";   int minLength = Math.min(str1.length(), str2.length());  boolean areEqual = true;   for (int i = 0; i < minLength; i++) {  if (str1.charAt(i) != str2.charAt(i)) {  areEqual = false;  if (str1.charAt(i) < str2.charAt(i)) {  System.out.println("\"" + str1 + "\" comes before \"" + str2 + "\" in lexicographical order");  } else {  System.out.println("\"" + str2 + "\" comes before \"" + str1 + "\" in lexicographical order");  }  break;  }  }   if (areEqual) {  if (str1.length() < str2.length()) {  System.out.println("\"" + str1 + "\" comes before \"" + str2 + "\" in lexicographical order");  } else if (str1.length() > str2.length()) {  System.out.println("\"" + str2 + "\" comes before \"" + str1 + "\" in lexicographical order");  } else {  System.out.println("Both strings are equal");  }  }  } }** |

**8. Compare Two Strings**

**Problem:**

**Write a Java program to compare two strings lexicographically (dictionary order) without using built-in compare methods.**

**Example Input:**

**String 1: "apple"**

**String 2: "banana"**

**Expected Output:**

**"apple" comes before "banana" in lexicographical order**

|  |
| --- |
| **public class Order {  public static void main(String[] args) {  String str1 = "apple";  String str2 = "banana";   int minLength = Math.min(str1.length(), str2.length());  boolean areEqual = true;   for (int i = 0; i < minLength; i++) {  if (str1.charAt(i) != str2.charAt(i)) {  areEqual = false;  if (str1.charAt(i) < str2.charAt(i)) {  System.out.println("\"" + str1 + "\" comes before \"" + str2 + "\" in lexicographical order");  } else {  System.out.println("\"" + str2 + "\" comes before \"" + str1 + "\" in lexicographical order");  }  break;  }  }   if (areEqual) {  if (str1.length() < str2.length()) {  System.out.println("\"" + str1 + "\" comes before \"" + str2 + "\" in lexicographical order");  } else if (str1.length() > str2.length()) {  System.out.println("\"" + str2 + "\" comes before \"" + str1 + "\" in lexicographical order");  } else {  System.out.println("Both strings are equal");  }  }  } }** |

**9. Find the Most Frequent Character**

**Problem:**

**Write a Java program to find the most frequent character in a string. Example Input:**

**String: "success"**

**Expected Output:**

**Most Frequent Character: 's'**

|  |
| --- |
| **public class Repeted {  public static void main(String[] args) {  String input = "success";  System.out.println("Most Frequent Character: '" + findMostFrequentCharacter(input) + "'");  }   public static char findMostFrequentCharacter(String str) {  int[] freq = new int[256];  for (char c : str.toCharArray()) freq[c]++;  char result = 0;  int max = 0;  for (char c : str.toCharArray()) {  if (freq[c] > max) {  max = freq[c];  result = c;  }  }  return result;  } }** |

**10. Remove a Specific Character from a String**

**Problem:**

**Write a Java program to remove all occurrences of a specific character from a string. Example Input:**

**String: "Hello World"**

**Character to Remove: 'l'**

**Expected Output:**

**Modified String: "Heo Word"**

|  |
| --- |
| **public class Remove {  public static void main(String[] args) {  String input = "Hello World";  char charToRemove = 'l';  String result = input.replace(Character.toString(charToRemove), "");  System.out.println("Modified String: " + result);  } }** |

**11. Write a Java program that accepts two strings from the user and checks if the two strings are anagrams of each other (i.e., whether they contain the same characters in any order).**

|  |
| --- |
| **import java.util.Arrays; import java.util.Scanner;  public class Anagrams {  public static void main(String[] args) {  Scanner scanner = new Scanner(System.in);   System.out.print("Enter the first string: ");  String str1 = scanner.nextLine();   System.out.print("Enter the second string: ");  String str2 = scanner.nextLine();   if (areAnagrams(str1, str2)) {  System.out.println("The strings are anagrams.");  } else {  System.out.println("The strings are not anagrams.");  }  }   private static boolean areAnagrams(String str1, String str2) {  char[] arr1 = str1.replaceAll("\\s", "").toLowerCase().toCharArray();  char[] arr2 = str2.replaceAll("\\s", "").toLowerCase().toCharArray();   Arrays.sort(arr1);  Arrays.sort(arr2);   return Arrays.equals(arr1, arr2);  } }** |

**12. Write a replace method in Java that replaces a given word with another word in a sentence:**

|  |
| --- |
| **public class Replace {  public static String replaceWord(String sentence, String target, String replacement) {  return sentence.replace(target, replacement);  }   public static void main(String[] args) {  String sentence = "The quick brown fox jumps over the lazy dog.";  String result = replaceWord(sentence, "fox", "cat");  System.out.println(result); }}** |