String(Module-24)
Assignment

Q.1 Program:

```
   package pw_java;
public class Remove_Dup {
            public static void main(String[] args) {
               String input = "programming"; // Example input string with duplicates
               System.out.println("Original String: " + input);
               // Call the method to remove duplicates
               String result = removeDuplicates(input);
               // Print the result
               System.out.println("String after removing duplicates: " + result);
            }
            // Method to remove duplicate characters from a string
            public static String removeDuplicates(String str) {
               // Create a boolean array to track seen characters
               boolean[] seen = new boolean[256]; // Supports ASCII characters
               StringBuilder result = new StringBuilder();
               // Iterate through each character in the string
               for (int i = 0; i < str.length(); i++) {
                  char c = str.charAt(i);
                  // If the character hasn't been seen, add it to the result
                  if (!seen[c]) {
                     result.append(c);
                     seen[c] = true; // Mark the character as seen
                  }
               }
               return result.toString();
            }
         }
```

Q.2 Program;

```
     public class Duplicate_Character {
          public static void main(String[] args) {
     String input = "programming"; // Example input string
     System.out.println("Input String: " + input);
     // Call the method to print duplicate characters
     printDuplicateCharacters(input);
  }
  public static void printDuplicateCharacters(String str) {
     char[] chars = str.toCharArray(); // Convert string to a character array
     boolean[] visited = new boolean[256]; // Array to track visited characters (for ASCII)

     System.out.println("Duplicate Characters:");
     for (int i = 0; i < chars.length; i++) {
```

```
        if (!visited[chars[i]]) { // If the character is not already visited
            int count = 0;
            for (int j = i; j < chars.length; j++) {
                if (chars[i] == chars[j]) {
                    count++; // Count occurrences of the character
                }
            }
            if (count > 1) { // If the character appears more than once, it's a duplicate
                System.out.println(chars[i] + " - " + count + " times");
            }
            visited[chars[i]] = true; // Mark the character as visited
        }
    }
  }
}
```

Q.3 Program:

```
  package pw_java;
public class palindrome_check {
        public static void main(String[] args) {
            String input = "2552"; // Input string
            System.out.println("Input String: " + input);
            // Check if the input is a palindrome
            if (isPalindrome(input)) {
                System.out.println(input + " is a palindrome.");
            } else {
                System.out.println(input + " is not a palindrome.");
            }
        }
        public static boolean isPalindrome(String str) {
            int start = 0;              // Start pointer
            int end = str.length() - 1;    // End pointer
            // Compare characters from both ends
            while (start < end) {
                if (str.charAt(start) != str.charAt(end)) {
                    return false; // If characters don't match, it's not a palindrome
                }
                start++;
                end--;
            }
            return true; // If all characters match, it's a palindrome
        }
    }
```

Q.4 Program:

```
  package pw_java;
public class Count_Vowel_Cono_SpecChar {
        public static void main(String[] args) {
```

```
                String inputString = "abcoui@yuAziyu$";
                int vowels = 0;
                int consonants = 0;
                int specialChars = 0;
                // Convert the string to lowercase for easier vowel checking
                inputString = inputString.toLowerCase();
                for (int i = 0; i < inputString.length(); i++) {
                    char ch = inputString.charAt(i);
                    if (Character.isLetter(ch)) { // Check if it's a letter
                        if (ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u') {
                            vowels++;
                        } else {
                            consonants++;
                        }
                    } else if (!Character.isWhitespace(ch) && !Character.isDigit(ch)) { // Check if it's not
a whitespace or digit
                        specialChars++;
                    }
                }
                System.out.println("Vowels: " + vowels);
                System.out.println("Consonants: " + consonants);
                System.out.println("Special Characters: " + specialChars);

            }
}
```

Q.5 Program:

```
  package pw_java;
import java.util.Scanner;
public class AnagramChecker {
  public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    System.out.print("Enter the first string: ");
    String str1 = scanner.nextLine().toLowerCase(); // Convert to lowercase
    System.out.print("Enter the second string: ");
    String str2 = scanner.nextLine().toLowerCase(); // Convert to lowercase
    if (areAnagrams(str1, str2)) {
      System.out.println(str1 + " and " + str2 + " are anagrams.");
    } else {
      System.out.println(str1 + " and " + str2 + " are not anagrams.");
    }
    scanner.close();
  }
  public static boolean areAnagrams(String str1, String str2) {
    // Remove whitespace and check lengths first for quick optimization
    str1 = str1.replaceAll("\\s", "");
    str2 = str2.replaceAll("\\s", "");
    if (str1.length() != str2.length()) {
```

```
        return false; // If lengths are different, they can't be anagrams
    }
    // Use character counting (most efficient with minimal built-in functions)
    int[] charCount = new int[256]; // Assuming ASCII characters
    for (int i = 0; i < str1.length(); i++) {
        charCount[str1.charAt(i)]++;
        charCount[str2.charAt(i)]--;
    }
    for (int count : charCount) {
        if (count != 0) {
            return false; // If any count is not zero, they are not anagrams
        }
    }
    return true; // All counts are zero, they are anagrams
  }
}
```

Q.6 Program:

```
  package pw_java;
import java.util.Scanner;
public class PangramChecker {
  public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    System.out.print("Enter a string: ");
    String input = scanner.nextLine().toLowerCase(); // Convert to lowercase
    if (isPangram(input)) {
        System.out.println("\"" + input + "\" is a pangram.");
    } else {
        System.out.println("\"" + input + "\" is not a pangram.");
    }
    scanner.close();
  }
  public static boolean isPangram(String str) {
    // Use a boolean array to track the presence of each letter (a-z)
    boolean[] alphabetPresent = new boolean[26];
    // Iterate through the string
    for (int i = 0; i < str.length(); i++) {
        char ch = str.charAt(i);
        if ('a' <= ch && ch <= 'z') { // Check if it's a lowercase letter
            alphabetPresent[ch - 'a'] = true; // Mark the corresponding letter as present
        }
    }
    // Check if all letters (a-z) are present
    for (boolean present : alphabetPresent) {
        if (!present) {
            return false; // If any letter is missing, it's not a pangram
        }
    }
```

```
        return true; // All letters are present, it's a pangram
    }
}
```

Q.7 program:

```
    package pw_java;
import java.util.Scanner;
public class UniqueCharacters {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a string: ");
        String inputString = scanner.nextLine();
        if (hasUniqueCharacters(inputString)) {
            System.out.println("The string \"" + inputString + "\" has all unique characters.");
        } else {
            System.out.println("The string \"" + inputString + "\" does not have all unique characters.");
        }
        scanner.close();
    }
    public static boolean hasUniqueCharacters(String str) {
        if (str == null || str.isEmpty()) {
            return true; // Empty string or null is considered to have unique characters
        }
        // Optimization: If string length is greater than 256 (or 128 for ASCII)
        // it cannot have all unique characters
        if (str.length() > 256) { // Assuming extended ASCII, use 128 for basic ASCII
            return false;
        }
        boolean[] charSet = new boolean[256]; // Array to track character presence
            }
            charSet[val] = true; // Mark character as present
        }
        return true; // All characters are unique
    }
}
```

Q.8 Program

```
    package pw_java;
import java.util.Scanner;
public class MaxOccurringChar {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a string: ");
        String inputString = scanner.nextLine();
        scanner.close(); // Close the scanner
        if (inputString == null || inputString.isEmpty()) {
            System.out.println("Input string is empty or null.");
            return; // Exit if the string is empty or null
```

```
        }
        char maxChar = findMaxOccurringChar(inputString);
        System.out.println("Maximum occurring character: " + maxChar);
    }
    public static char findMaxOccurringChar(String str) {
        int[] charCounts = new int[256]; // Assuming extended ASCII
        for (int i = 0; i < str.length(); i++) {
            charCounts[str.charAt(i)]++;
        }
        int maxCount = 0;
        char maxChar = '\0'; // Initialize with null character
        for (int i = 0; i < charCounts.length; i++) {
            if (charCounts[i] > maxCount) {
                maxCount = charCounts[i];
                maxChar = (char) i;
            }
        }
        return maxChar;
    }
}
```