1. WAP(Write a Program) to remove Duplicates from a String.(Take any String example with duplicates character)

```
public class Main {
    public static void main(String[] args) {
        String str = "programming";
        String result = "";

        for (char ch : str.toCharArray()) {
            if (result.indexOf(ch) == -1) { // Check if the character is not already in the result
                result += ch;
            }
        }

        System.out.println("String after removing duplicates: " + result);
    }
}
```

2. WAP to print Duplicates characters from the String

```
import java.util.HashSet;

public class Main {
    public static void main(String[] args) {
        String str = "programming";
        HashSet<Character> seen = new HashSet<>();
        System.out.println("Duplicate characters:");

        for (char ch : str.toCharArray()) {
            if (!seen.add(ch)) { // If already in 'seen', it's a duplicate
                System.out.println(ch);
            }
        }
    }
}
```

3. WAP to check if “2552” is palindrome or note

```
public class Main {
    public static void main(String[] args) {
        String str = "2552";
        String reversed = new StringBuilder(str).reverse().toString();

        if (str.equals(reversed)) {
            System.out.println(str + " is a palindrome.");
        } else {
            System.out.println(str + " is not a palindrome.");
        }
    }
}
```

4. WAP to count the number of consonants, vowels, special characters in a String.

```
public class Main {

    public static void main(String[] args) {
        String str = "Hello, have a good day!";
        int vowels = 0, consonants = 0, specialChars = 0;

        for (char ch : str.toCharArray()) {
            if (Character.isLetter(ch)) {
                if ("AEIOUaeiou".indexOf(ch) != -1) {
                    vowels++;
                } else {
                    consonants++;
                }
            } else if (!Character.isWhitespace(ch)) {
                specialChars++;
            }
```

```
        }

        System.out.println("Vowels: " + vowels);
        System.out.println("Consonants: " + consonants);
        System.out.println("Special characters: " + specialChars);
    }
}
```

5. WAP to implement Anagram Checking least inbuilt methods being used.

```
import java.util.Arrays;

public class Main{
    public static void main(String[] args) {
        String str1 = "listen";
        String str2 = "silent";

        if (isAnagram(str1, str2)) {
            System.out.println("The strings are anagrams.");
        } else {
            System.out.println("The strings are not anagrams.");
        }
    }

    public static boolean isAnagram(String str1, String str2) {
        if (str1.length() != str2.length()) {
            return false;
        }

        char[] arr1 = str1.toCharArray();
        char[] arr2 = str2.toCharArray();
        Arrays.sort(arr1);
        Arrays.sort(arr2);

        return Arrays.equals(arr1, arr2);
    }
}
```

6. WAP to implement Pangram Checking with least inbuilt methods being used.

```
public class Main {
    public static void main(String[] args) {
        String str = "The quick brown fox jumps over the lazy dog";
        if (isPangram(str)) {
            System.out.println("The string is a pangram.");
        } else {
            System.out.println("The string is not a pangram.");
        }
    }

    public static boolean isPangram(String str) {
        boolean[] alphabet = new boolean[26];
        int index;

        for (char ch : str.toLowerCase().toCharArray()) {
            if (ch >= 'a' && ch <= 'z') {
                index = ch - 'a';
                alphabet[index] = true;
            }
        }

        for (boolean flag : alphabet) {
            if (!flag) {
                return false;
            }
        }

        return true;
    }
}
```

7. WAP to find if String contains all unique characters.

```
import java.util.HashMap;

public class Main {
    public static void main(String[] args) {
        String str = "abcdef";
        HashMap<Character, Integer> map = new HashMap<>();
        boolean isUnique = true;

        for (char ch : str.toCharArray()) {
            if (map.containsKey(ch)) {
                isUnique = false;
                break;
            }
            map.put(ch, 1);
        }

        System.out.println("The string has all unique characters: " + isUnique);
    }
}
```

8. WAP to find the maximum occurring character in a String.

```
import java.util.HashMap;

public class MaxOccurringCharacter {
    public static void main(String[] args) {
        String str = "sampleprogram";
        HashMap<Character, Integer> map = new HashMap<>();

        for (char ch : str.toCharArray()) {
            map.put(ch, map.getOrDefault(ch, 0) + 1);
        }

        char maxChar = '\0';
        int maxCount = 0;

        for (char ch : map.keySet()) {
```

```
            if (map.get(ch) > maxCount) {
                maxCount = map.get(ch);
                maxChar = ch;
            }
        }

        System.out.println("Maximum occurring character: " + maxChar + " (occurs "
+ maxCount + " times)");
    }
}
```