ASSIGNMENT-13(5FEB)

Q. WAP(Write a program) to remove Duplicates from a string.(Take any String Example with duplicates character).

Ans-

```java
public class String_duplicate {
    public static void main(String[] args) {
        String str = "Amit Ranjan";
        String result = removeDuplicates(str);
        System.out.println(result);
    }

    public static String removeDuplicates(String str) {
        StringBuilder sb = new StringBuilder();
        for (int i = 0; i < str.length(); i++) {
            char c = str.charAt(i);
            if (sb.indexOf(String.valueOf(c)) == -1) {
                sb.append(c);
            }
        }
        return sb.toString();
    }
}
```

Q. WAP to print Duplicate characters from a string.

Ans-

```java
public class PrintDuplicates {
    public static void main(String[] args) {
        String str = "hello world"; // Example string with duplicates
        printDuplicates(str);
    }

    public static void printDuplicates(String str) {
        // Convert the string to lowercase to treat uppercase and lowercase
characters as duplicates
        str = str.toLowerCase();
        // Create an array to keep track of character frequencies
        int[] frequency = new int[26];
        // Iterate over each character in the string and update the frequency
array
        for (int i = 0; i < str.length(); i++) {
            char c = str.charAt(i);
            frequency[c - 'a']++;
        }
        // Iterate over the frequency array and print any character that 
appears more than once
        for (int i = 0; i < frequency.length; i++) {
            if (frequency[i] > 1) {
                System.out.println((char) (i + 'a') + " appears " + 
frequency[i] + " times.");
            }
        }
    }
}
```

Q. WAP to check if 2552 is a palindrome or not?

Ans-

```
public class check_palindrome {
    public static void main(String[] args) {
        String str="2552";
        String str2="";
        for (int i = str.length()-1; i >=0; i--) {
            str2=str2+str.charAt(i);
        }
        if (str.equals(str2)) {
            System.out.println("It is a palindrome");
        }else{
            System.out.println("It is not a palindrome");
        }

    }
}
```

Q. WAP to count the number of consonants , vowels, special characters in a string.

Ans-

```
public class count_characters {
    public static void main(String[] args) {
        String str = "Amit Ranjan";
        countCharacters(str);
    }

    public static void countCharacters(String str) {
        int vowels = 0;
        int consonants = 0;
        int specials = 0;
        str = str.toLowerCase();
        for (int i = 0; i < str.length(); i++) {
            char c = str.charAt(i);
            if (c >= 'a' && c <= 'z') {
                if (c == 'a' || c == 'e' || c == 'i' || c == 'o' || c == 'u')
{
                    vowels++;
                } else {
                    consonants++;
                }
            } else if (c >= '0' && c <= '9') {

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

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

Ans-

```
public class Anagram_check {
    public static void main(String[] args) {
        String str1 = "listen";
        String str2 = "silent";
        boolean isAnagram = checkAnagram(str1, str2);
        if (isAnagram) {
            System.out.println(str1 + " and " + str2 + " are anagrams.");
        } else {
            System.out.println(str1 + " and " + str2 + " are not anagrams.");
        }
    }

    public static boolean checkAnagram(String str1, String str2) {
        // Convert the strings to lowercase to ignore case
        str1 = str1.toLowerCase();
        str2 = str2.toLowerCase();

        // Check if the lengths of the two strings are the same
        if (str1.length() != str2.length()) {
            return false;
        }
        // Create arrays to keep track of the frequency of each character in
the strings
        int[] freq1 = new int[26];
        int[] freq2 = new int[26];

        // Iterate over each character in the two strings and update the
frequency arrays
        for (int i = 0; i < str1.length(); i++) {
            freq1[str1.charAt(i) - 'a']++;
            freq2[str2.charAt(i) - 'a']++;
        }

        // Check if the frequency arrays are the same
        for (int i = 0; i < 26; i++) {
            if (freq1[i] != freq2[i]) {
                return false;
            }
        }

        // If we get here, the two strings are anagrams
        return true;
    }
}
```

Q. WAP to implement Pangram checking with least inbuilt methods being used.

Ans-

```
import java.util.Scanner;

public class pangram {
    public static void main(String[] args) {
        boolean flag=false;
        Scanner sc= new Scanner(System.in);
        System.out.print("Enter Your String: ");
        String str=sc.nextLine();
        str=str.replace(" ", "");
        str=str.toUpperCase();
        char ch[]=str.toCharArray();
        int[] arr= new int[26];
        for (int i = 0; i < ch.length; i++) {
            arr[ch[i]-65]++;
        }
        for (int i = 0; i < arr.length; i++) {
            if (arr[i]==0) {
                System.out.println("It is not a pangram");
                flag=true;
                break;
            }
        }
        if (flag==false) {
            System.out.println("It is a pangram");
        }
    }
}
```

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

Ans-

```
public class check_special_char{
    public static void main(String[] args) {
        String str = "abcdefgg";
        boolean hasUniqueChars = checkUniqueChars(str);
        if (hasUniqueChars) {
            System.out.println(str + " contains all unique characters.");
        } else {
            System.out.println(str + " does not contain all unique
characters.");
        }
    }

    public static boolean checkUniqueChars(String str) {
        // Create a boolean array to keep track of the occurrence of each
character
        boolean[] charSet = new boolean[256];

        // Iterate over each character in the string and check if it has
already occurred
        for (int i = 0; i < str.length(); i++) {
            int val = str.charAt(i);
            if (charSet[val]) {
                return false;
            }
            charSet[val] = true;
        }

        // If we get here, the string contains all unique characters
        return true;
    }
}
```

Q. WAP to find the maximum occurring character in a string.

Ans-

```java
public class MaxChar {
    public static void main(String[] args) {
        String str = "Amit Ranjan";
        char maxChar = findMaxChar(str);
        System.out.println("The maximum occurring character in " + str + " is 
" + maxChar);
    }

    public static char findMaxChar(String str) {
        // Create an array to keep track of the frequency of each character
        int[] charFreq = new int[256];

        // Iterate over each character in the string and update the frequency 
array
        for (int i = 0; i < str.length(); i++) {
            charFreq[str.charAt(i)]++;
        }

        // Find the character with the highest frequency
        char maxChar = ' ';
        int maxFreq = 0;
        for (int i = 0; i < charFreq.length; i++) {
            if (charFreq[i] > maxFreq) {
                maxChar = (char) i;
                maxFreq = charFreq[i];
            }
        }

        return maxChar;
    }
}
```