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

```
public class Test {
    // This code covers the case of whitespaces between the strings and
keeps the location of whitespaces same as before
    public static void main(String[] args){
    // Case 1 - For all Lowercase letters
    String s1 = "java backend course";
    char ch1[] = s1.toCharArray();
    int arr1[] = new int[27];
    for(int i=0; i<ch1.length; ++i){
        if(ch1[i] == ' ')
            arr1[26]++;
        else
            arr1[ch1[i] - 97]++;
    }
    char nCh1[] = new char[50];
    int j1=0;
    for(int i=0; i<ch1.length; ++i){
        if(ch1[i] == ' ')
            nCh1[j1++] = ch1[i];
        else if(arr1[ch1[i]-97] < 2)
            nCh1[j1++] = ch1[i];
    }
    // This will print the String in char array format without any
duplicates
    System.out.println(nCh1);
    // Case 2 - For all Uppercase letters
    String s2 = "JAVA BACKEND COURSE";
    char ch2[] = s2.toCharArray();
    int arr2[] = new int[27];
    for(int i=0; i<ch2.length; ++i){
        if(ch2[i] == ' ')
            arr2[26]++;
        else
            arr2[ch2[i] - 65]++;
    }
    char nCh2[] = new char[50];
    int j2=0;
    for(int i=0; i<ch2.length; ++i){
        if(ch2[i] == ' ')
            nCh2[j2++] = ch2[i];
        else if(arr2[ch2[i]-65] < 2)
            nCh2[j2++] = ch2[i];
    }
    // This will print the String in char array format without any
duplicates
    System.out.println(nCh2);
    }
}
```

2. WAP to print Duplicates characters from the String.
Ans:

```
public class PrintDuplicates {
    // Considering strings with no white spaces
    public static void main(String[] args) {
        // Considering special case of all lowercase letters
        // Code for all uppercase is similar to this code, just
        // replace 97 with 65
        String s = "pwjavabackendcoursestarted";
        char ch[] = s.toCharArray();
        int arr[] = new int[26];
        for(int i=0; i< ch.length; ++i)
            arr[ch[i]-97]++;
        System.out.println("The duplicate characters are as follows: ");
        for(int i=0; i<ch.length; ++i){
            // if they are duplicates, then
            if(arr[ch[i]-97] > 1){
                // once we know that they are duplicates, print them and
set the array value to 0
                System.out.println(ch[i]);
                // to avoid repetition
                arr[ch[i]-97]=0;
            }
        }
    }
}
```

3. WAP to check if “2552” is palindrome or not.
Ans:

```
public class Test {
    public static void main(String[] args){
        String str1 = "2552";
        boolean isPalindrome = true;
        for(int i=0, j=str1.length()-1; i<str1.length() && j >= 0; ++i,
--j){
            if(str1.charAt(i) != str1.charAt(j)){
                isPalindrome = false;
            }
        }
        if (isPalindrome)
            System.out.println(str1 + " is a Palindrome");
        else
            System.out.println(str1 + " is not a Palindrome");
    }
}
```

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

```
public class NumbofConsoVowelsSpecialCharacters {
    // Considering a string with no spaces
    // Here, we are considering letters as vowels or consonants
    // and all other characters as special characters
    // ASCII values of A_Z and a-z are 65-90 and 97-122 respectively
    public static void main(String[] args) {
        String s1 = "pw java backend course is awesome!!!";
        char ch[] = s1.toCharArray();
        int numVowels = 0;             // Number of vowels
        int numConsonants = 0;         // Number of consonants
        int numSpecialCharacters = 0; // Number of special charactes
        for(int i=0; i<ch.length; ++i){
            int a = ch[i]; // Get the ASCII value of the given character
            if((a>= 65 && a<=90) || (a>=97 && a<=122)){
                if(ch[i]=='a' || ch[i]=='e' || ch[i]=='i' || ch[i]=='o' ||
ch[i]=='u' ||
                    ch[i]=='A' || ch[i]=='E' || ch[i]=='I' || ch[i]=='O' ||
ch[i]=='U')
                    numVowels++;
                else
                    numConsonants++;
            }
        }
        // remaining are special characters
        numSpecialCharacters = ch.length - numConsonants -numVowels;
        System.out.println("The number of vowels in the given String are "
+ numVowels);
        System.out.println("The number of consonants in the given String
are " + numConsonants);
        System.out.println("The number of special characters in the given
String are " + numSpecialCharacters);

    }
}
```

5. WAP to implement Anagram Checking with the least inbuilt methods being used.
Ans:
In both codes below, only one inbuilt method is used, toCharArray()
// Code 1 - Strings are Anagrams

```
public class AnagramLeastInbuiltMenthods {
    //Consider strings of all lowercase letters with no spaces
    public static void main(String[] args) {
        String s1 = "pwjava";
        String s2 = "ajavwp";
        char ch1[] = s1.toCharArray();
        char ch2[] = s2.toCharArray();
        boolean areAnagrams = true;
        int arr[] = new int[26];
        for(int i=0; i<ch1.length; ++i)
            arr[ch1[i]-97]++;
        for(int i=0; i<ch2.length; ++i)
            arr[ch2[i]-97]--;
        for(int i=0; i<26; ++i){
            if(arr[i] != 0){
                areAnagrams = false;
                break;
            }
        }
        if(areAnagrams)
            System.out.println("The given pair of Strings are Anagrams");
        else
            System.out.println("The given pair of Strings are not
Anagrams");
    }
}
```

// Code 2 - Strings are not Anagrams

```
public class AnagramLeastInbuiltMenthods {
    //Consider strings of all lowercase letters with no spaces
    public static void main(String[] args) {
        String s1 = "pwjaava";
        String s2 = "ajavvwp";
        char ch1[] = s1.toCharArray();
        char ch2[] = s2.toCharArray();
        boolean areAnagrams = true;
        int arr[] = new int[26];
        for(int i=0; i<ch1.length; ++i)
            arr[ch1[i]-97]++;
        for(int i=0; i<ch2.length; ++i)
            arr[ch2[i]-97]--;
        for(int i=0; i<26; ++i){
            if(arr[i] != 0){
                areAnagrams = false;
                break;
            }
        }
        if(areAnagrams)
            System.out.println("The given pair of Strings are Anagrams");
        else
            System.out.println("The given pair of Strings are not
Anagrams");
    }
}
```

6. WAP to implement Pangram Checking with the least inbuilt methods being used.
Ans:
// This code uses only 1 inbuilt method, indexOf(char)

```
public class PangramLeastInbuiltMenthods {
    // I have taken a String in all lowercase letters
    // to minimize use of inbuilt methods
       public static void main(String[] args) {
        String s1 = "the quick brown fox jumps over lazy dog";
        String s2 = "the quick brown fox jumps lazy dog";
        char compare1[] = new char[26];
        // First we check for S1 which is a Pangram
        boolean isS1Pangram = true;
        int a = 97;
        for(int i=0; i<compare1.length; ++i){
            compare1[i] = (char)a;
            int ans = s1.indexOf(compare1[i]);
            if(ans < 0)
                isS1Pangram = false;
            a++;
        }
        if(isS1Pangram)
            System.out.println("The first String is a Pangram");
        else
            System.out.println("The first string is not a Pangram");
        // Now we check for S2 which is not a Pangram
        boolean isS2Pangram = true;
        int b = 97;
        char compare2[] = new char[26];
        for(int i=0; i<compare2.length; ++i){
            compare2[i] = (char)b;
            int ans = s2.indexOf(compare2[i]);
            if(ans < 0)
                isS2Pangram = false;
            b++;
        }
        if(isS2Pangram)
            System.out.println("The second String is a Pangram");
        else
            System.out.println("The second string is not a Pangram");
    }
}
```

7. WAP to find if a String contains all unique characters.
Ans:

```
public class CheckUnique {
    // Consider the string as only made up of lowercase letters and there
are
    // no special characters or uppercase letters(for simplicity)
    // For String with all uppercase letters, replace 65 with 97 in the
below code
    public static void main(String[] args) {
        String s = "abcdefghij";
        char ch[] = s.toCharArray();
        int arr[] = new int[26];
        for(int i=0; i<ch.length; ++i)
            arr[ch[i]-97]++;
        boolean isUnique = true;
        for(int i=0; i<ch.length; ++i){
            if(arr[ch[i]-97]>1){
                isUnique =false;
                break;
            }
        }
        if(isUnique)
            System.out.println("The given string contains all unique
characters");
        else
        System.out.println("The given string does not contain all unique
characters");
    }
}
```

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

```
public class MaxOccuringChar {
    //Consider String has all lowercase letters
    //For all uppercase letters, replace 97 with 65
    public static void main(String[] args) {
        String s = "pwjavacourseisgoodforbackenddevelopers";
        char ch[] = s.toCharArray();
        int max = 0;
        int arr[] = new int[26];
        char ans='\u0000'; // Initialize a char with this default value
        for(int i=0; i<ch.length; ++i)
            arr[ch[i]-97]++;
        for(int i=0; i<ch.length; ++i)  {
            if(arr[ch[i]-97]> max){
                max = arr[ch[i]-97];
                ans = ch[i];
            }
        }
        System.out.println("The maximum occuring character in the given
string is " + ans);
    }
}
```