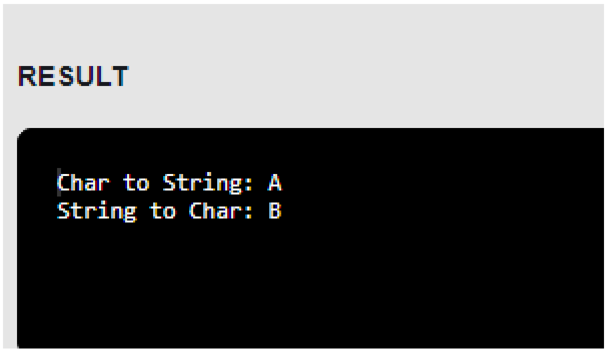
Java Assignment 6

Java Strings Programs

1. Java Program to Convert char to String and String to Char

```
public class CharToStringAndStringToChar {
    public static void main(String[] args) {
        // Convert char to String
        char myChar = 'A';
        String charToString = String.valueOf(myChar);
        System.out.println("Char to String: " +
charToString);

        // Convert String to char
        String myString = "B";
        if (myString.length() == 1) {
            char stringToChar = myString.charAt(0);
            System.out.println("String to Char: " +
stringToChar);
        } else {
            System.out.println("Input String is not a
single character.");
        }
    }
}
```

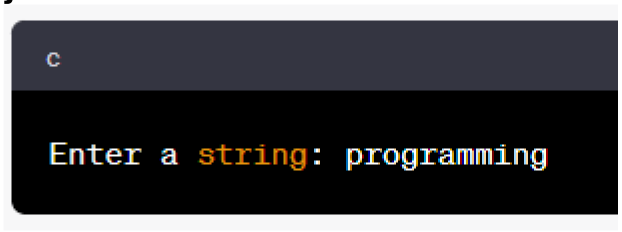


RESULT

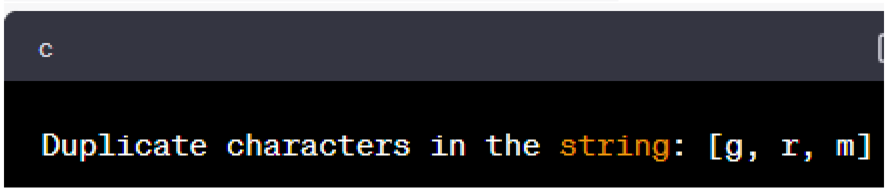
```
Char to String: A
String to Char: B
```

2. Java Program to find duplicate characters in a String

```
import java.util.HashSet;
import java.util.Scanner;
import java.util.Set;

public class FindDuplicateCharacters {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a string: ");
        String input = scanner.nextLine();
        scanner.close();

        Set<Character> uniqueCharacters = new HashSet<>();
        Set<Character> duplicateCharacters = new HashSet<>();

        for (char ch : input.toCharArray()) {
            if (!uniqueCharacters.add(ch)) {
                duplicateCharacters.add(ch);
            }
        }

        System.out.println("Duplicate characters in the string: " +
    duplicateCharacters);
    }
}
```



```
c

Enter a string: programming
```

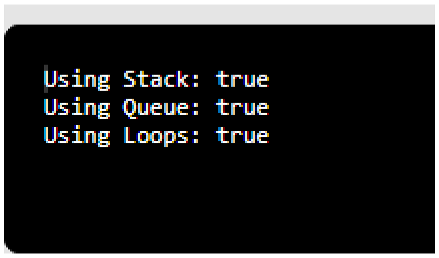


```
c

Duplicate characters in the string: [g, r, m]
```

3. **Java Program to check Palindrome String using Stack, Queue, For and While loop**

```
import java.util.LinkedList;
import java.util.Queue;
import java.util.Stack;

public class PalindromeCheck {
    public static void main(String[] args) {
        String input = "racecar"; // Change this to the string you want to
check

        // Using a stack
        boolean isPalindromeWithStack =
checkPalindromeWithStack(input);
        System.out.println("Using Stack: " + isPalindromeWithStack);

        // Using a queue
        boolean isPalindromeWithQueue =
checkPalindromeWithQueue(input);
```

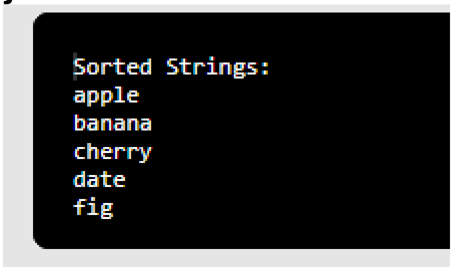
```
    }

        System.out.println("Using Queue: " + isPalindromeWithQueue);

        // Using loops
        boolean isPalindromeWithLoops =
checkPalindromeWithLoops(input);
        System.out.println("Using Loops: " + isPalindromeWithLoops);
    }

    // Check palindrome using a stack
    public static boolean checkPalindromeWithStack(String input) {
        Stack<Character> stack = new Stack<>();
        for (char c : input.toCharArray()) {
            stack.push(c);
        }

        StringBuilder reversed = new StringBuilder();
        while (!stack.isEmpty()) {
            reversed.append(stack.pop());
        }

        return input.equals(reversed.toString());
    }

    // Check palindrome using a queue
    public static boolean checkPalindromeWithQueue(String input) {
        Queue<Character> queue = new LinkedList<>();
        for (char c : input.toCharArray()) {
            queue.offer(c);
        }

        StringBuilder reversed = new StringBuilder();
        while (!queue.isEmpty()) {
            reversed.append(queue.poll());
        }

        return input.equals(reversed.toString());
    }

    // Check palindrome using loops
    public static boolean checkPalindromeWithLoops(String input) {
        int left = 0;
        int right = input.length() - 1;

        while (left < right) {
            if (input.charAt(left) != input.charAt(right)) {
                return false;
            }
            left++;
            right--;
        }

        return true;
    }
}
```



```
Using Stack: true
Using Queue: true
Using Loops: true
```

4. Java Program to sort strings in alphabetical order

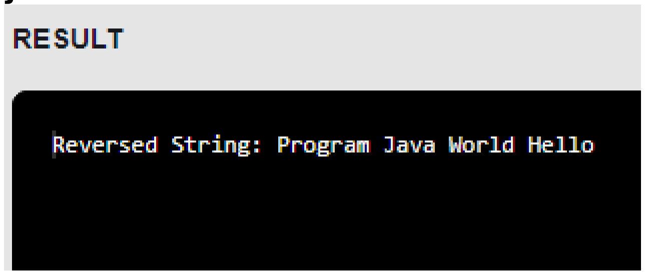
```
import java.util.Arrays;

public class SortStringsAlphabetically {
    public static void main(String[] args) {
        String[] strings = {"apple", "banana", "cherry", "date", "fig"};

        Arrays.sort(strings);

        System.out.println("Sorted Strings:");
        for (String str : strings) {
            System.out.println(str);
        }
    }
}
```



```
Sorted Strings:
apple
banana
cherry
date
fig
```

5. Java Program to reverse words in a String

```
public class ReverseWordsInString {
    public static void main(String[] args) {
        String input = "Hello World Java Program";

        String[] words = input.split(" ");
        StringBuilder reversedString = new StringBuilder();

        for (int i = words.length - 1; i >= 0; i--) {
            reversedString.append(words[i]);
            if (i > 0) {
                reversedString.append(" ");
            }
        }

        System.out.println("Reversed String: " + reversedString.toString());
    }
}
```

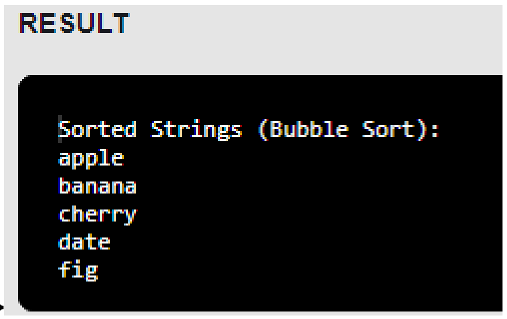


RESULT

```
Reversed String: Program Java World Hello
```

6. Java Program to perform bubble sort on Strings

```
import java.util.Arrays;

public class BubbleSortStrings {
    public static void main(String[] args) {
        String[] strings = {"apple", "banana", "cherry", "date", "fig"};

        for (int i = 0; i < strings.length - 1; i++) {
            for (int j = 0; j < strings.length - i - 1; j++) {
                if (strings[j].compareTo(strings[j + 1]) > 0) {
                    String temp = strings[j];
                    strings[j] = strings[j + 1];
                    strings[j + 1] = temp;
                }
            }
        }

        System.out.println("Sorted Strings (Bubble Sort):");
        for (String str : strings) {
            System.out.println(str);
        }
    }
```



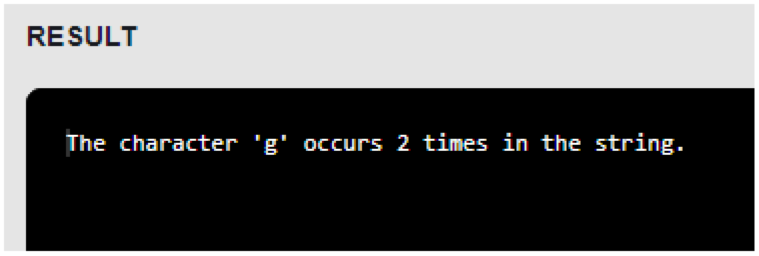
RESULT

```
Sorted Strings (Bubble Sort):
apple
banana
cherry
date
fig
```

```
}
```

7. Java program to find occurrence of a character in a String

```
public class CharacterOccurrenceInString {
    public static void main(String[] args) {
        String input = "programming";
        char targetChar = 'g';

        int count = 0;

        for (int i = 0; i < input.length(); i++) {
            if (input.charAt(i) == targetChar) {
                count++;
            }
        }

        System.out.println("The character '" + targetChar + "' occurs " +
count + " times in the string.");
    }
}
```

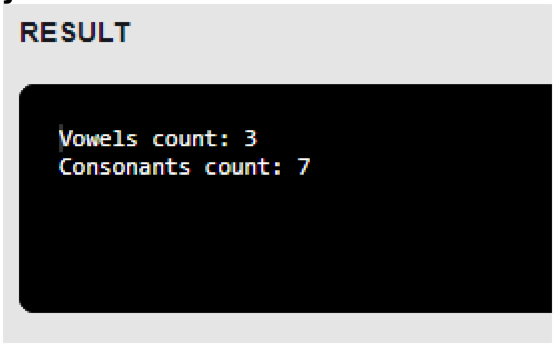


RESULT

```
The character 'g' occurs 2 times in the string.
```

8. Java program to count vowels and consonants in a String

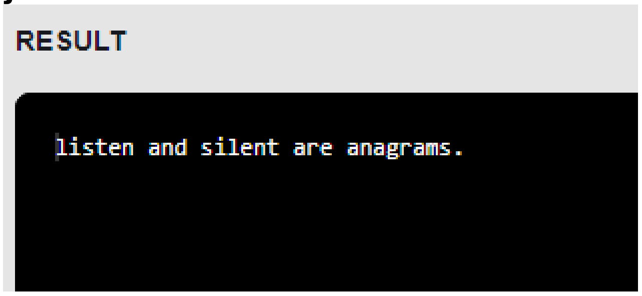
```
public class VowelsAndConsonantsCount {
    public static void main(String[] args) {
        String input = "Hello World";

        int vowelCount = 0;
        int consonantCount = 0;

        input = input.toLowerCase();

        for (int i = 0; i < input.length(); i++) {
            char ch = input.charAt(i);
            if (ch >= 'a' && ch <= 'z') {
                if (ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u') {
                    vowelCount++;
                } else {
                    consonantCount++;
                }
            }
        }

        System.out.println("Vowels count: " + vowelCount);
        System.out.println("Consonants count: " + consonantCount);
    }
}
```


RESULT

```
Vowels count: 3
Consonants count: 7
```

9. Java Program to check two strings are anagram or not

```
import java.util.Arrays;

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

        boolean areAnagrams = checkAnagrams(str1, str2);

        if (areAnagrams) {
            System.out.println(str1 + " and " + str2 + " are anagrams.");
        } else {
            System.out.println(str1 + " and " + str2 + " are not anagrams.");
        }
    }

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

        char[] charArray1 = str1.toCharArray();
        char[] charArray2 = str2.toCharArray();
        Arrays.sort(charArray1);
        Arrays.sort(charArray2);

        return Arrays.equals(charArray1, charArray2);
    }
}
```

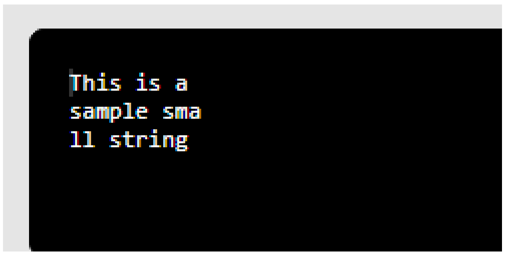


RESULT

```
listen and silent are anagrams.
```

10. Java Program to divide a string in ‘n’ equal parts

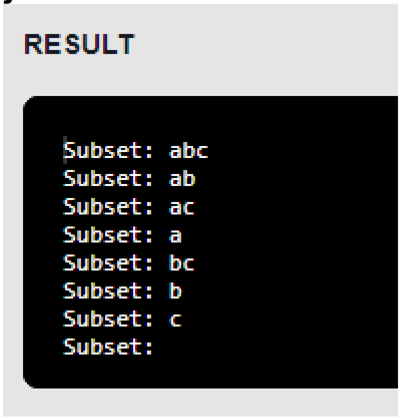
```
public class DivideStringIntoParts {
    public static void main(String[] args) {
        String input = "This is a sample small string";
        int n = 3;

        String[] parts = divideString(input, n);

        if (parts != null) {
            for (String part : parts) {
                System.out.println(part);
            }
        } else {
            System.out.println("String divided as equally as possible into " + n
+ " parts.");
        }
    }

    public static String[] divideString(String input, int n) {
        int length = input.length();
        int partLength = (int) Math.ceil((double) length / n);  // Calculate part
length, rounding up

        String[] parts = new String[n];

        for (int i = 0; i < n; i++) {
            int startIndex = i * partLength;
            int endIndex = Math.min((i + 1) * partLength, length);
            parts[i] = input.substring(startIndex, endIndex);
        }

        return parts;
    }
}
```



```
This is a
sample sma
ll string
```

11. **Java Program to find all subsets of a string**

```
public class StringSubsets {
    public static void main(String[] args) {
        String input = "abc";
        generateSubsets(input, 0, "");
    }

    public static void generateSubsets(String input, int index, String
currentSubset) {
        int n = input.length();

        if (index == n) {
            System.out.println("Subset: " + currentSubset);
            return;
        }

        // Include the current character in the subset
        generateSubsets(input, index + 1, currentSubset +
input.charAt(index));

        // Exclude the current character from the subset
        generateSubsets(input, index + 1, currentSubset);
    }
}
```

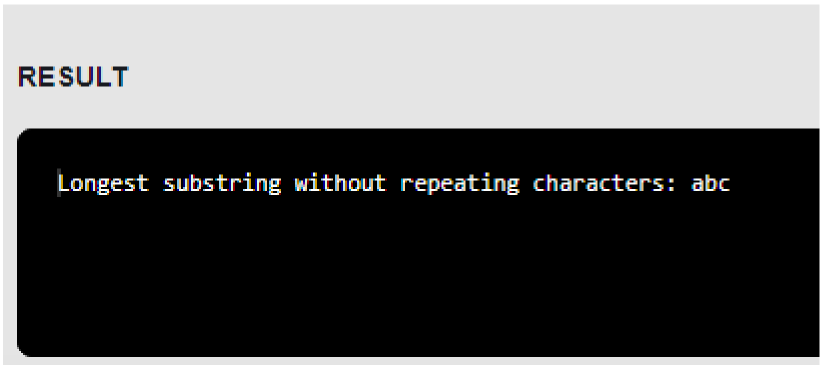


RESULT

```
Subset: abc
Subset: ab
Subset: ac
Subset: a
Subset: bc
Subset: b
Subset: c
Subset:
```

12. **Java Program to find longest substring without repeating characters**

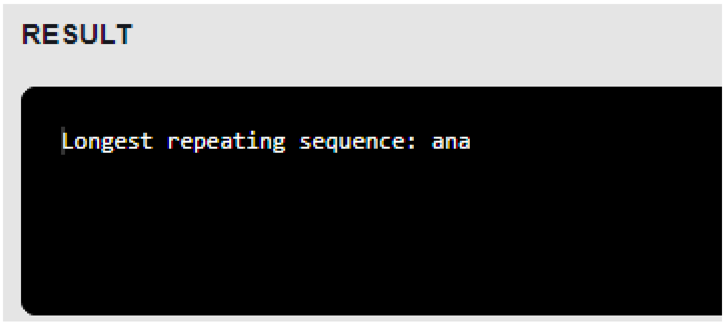
```
import java.util.HashMap;

public class LongestSubstringWithoutRepeatingChars {
    public static void main(String[] args) {
        String input = "abcabcbb";

        String longestSubstring = findLongestSubstring(input);

        System.out.println("Longest substring without repeating
characters: " + longestSubstring);
    }

    public static String findLongestSubstring(String input) {
        int n = input.length();
        int maxLength = 0;
        int start = 0;

        HashMap<Character, Integer> charIndexMap = new
HashMap<>();
        int maxStart = 0;

        for (int i = 0; i < n; i++) {
            char currentChar = input.charAt(i);

            if (charIndexMap.containsKey(currentChar) &&
charIndexMap.get(currentChar) >= start) {
                start = charIndexMap.get(currentChar) + 1;
            }

            charIndexMap.put(currentChar, i);

            if (i - start + 1 > maxLength) {
                maxLength = i - start + 1;
                maxStart = start;
            }
        }

        return input.substring(maxStart, maxStart + maxLength);
    }
}
```


RESULT

```
Longest substring without repeating characters: abc
```

13. **Java Program to find longest repeating sequence in a string**

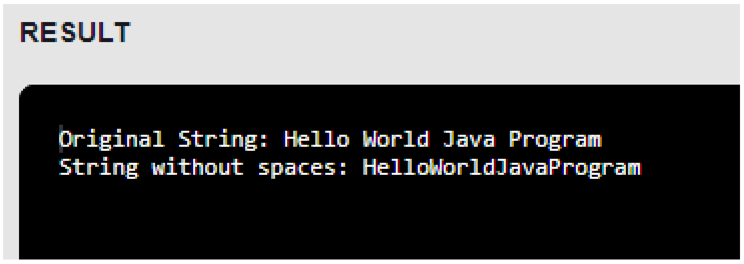
```
public class LongestRepeatingSequence {
   public static void main(String[] args) {
      String input = "banana";

      String longestRepeatingSequence =
findLongestRepeatingSequence(input);

      System.out.println("Longest repeating sequence: " +
longestRepeatingSequence);
   }

   public static String findLongestRepeatingSequence(String input) {
      int n = input.length();
      String longestSequence = "";

      for (int i = 0; i < n; i++) {
         for (int j = i + 1; j < n; j++) {
            int k = 0;
            while (j + k < n && input.charAt(i + k) == input.charAt(j + k)) {
               k++;
            }

            if (k > 0 && k > longestSequence.length()) {
               longestSequence = input.substring(i, i + k);
            }
         }
      }

      return longestSequence;
   }
}
```


RESULT

```
Longest repeating sequence: ana
```

14. **Java Program to remove all the white spaces from a string**

```
public class RemoveWhiteSpace {
    public static void main(String[] args) {
        String input = "Hello World Java Program";

        String stringWithoutSpaces = removeSpaces(input);

        System.out.println("Original String: " + input);
        System.out.println("String without spaces: " + stringWithoutSpaces);
    }

    public static String removeSpaces(String input) {
        // Use the regular expression "\\s" to match all white spaces
        return input.replaceAll("\\s", "");
    }
}
```


RESULT

```
Original String: Hello World Java Program
String without spaces: HelloWorldJavaProgram
```