100+ String

Tuesday, July 30, 2024 6:21 PM

Chapter 1: Basic String Operations

1. Reverse a String

java

```
public class ReverseString {
   public static void main(String[] args) {
      String str = "Hello";
      String reversed = new StringBuilder(str).reverse().toString();
      System.out.println("Reversed String: " + reversed);
   }
}
```

Output:

Reversed String: olleH

Other Method

33. Reverse Each Word in a String

java

```
public class ReverseWords {
   public static void main(String[] args) {
      String str = "Hello World";
      String[] words = str.split("\\s+");
      StringBuilder reversed = new StringBuilder();
      for (String word : words) {
         reversed.append(new
StringBuilder(word).reverse().toString()).append(" ");
      }
      System.out.println("Reversed Words: " +
reversed.toString().trim());
   }
}
```

Output:

Reversed Words: olleH dlroW

2. Check if a String is Palindrome

java

```
public class PalindromeCheck {
   public static void main(String[] args) {
      String str = "madam";
      String reversed = new StringBuilder(str).reverse().toString();
      boolean isPalindrome = str.equals(reversed);
      System.out.println("Is Palindrome: " + isPalindrome);
   }
}
```

Output:

Is Palindrome: true

Palindrome 2

```
publicclassPalendrom2{
publicstaticvoidmain(String[]args){
Stringstr="madam";
char[]charArray=str.toCharArray();
booleanisPalindrome=true;
intstart=0;
intend=charArray.length-1;
while(start<end){
if(charArray[start]!=charArray[end]){
isPalindrome=false;
break;
}
start++;
end--;
}
```

53. Find the Longest Palindromic Subsequence

```
public class LongestPalindromicSubsequence {
   public static void main(String[] args) {
      String str = "bbbab";
      int length = longestPalindromeSubseq(str);
      System.out.println("Length of Longest Palindromic Subsequence: " +
length);
   }
public static int longestPalindromeSubseq(String s) {
      int n = s.length();
      int[][] dp = new int[n][n];
      for (int i = n - 1; i >= 0; i--) {
         dp[i][i] = 1;
         for (int j = i + 1; j < n; j++) {
            if (s.charAt(i) == s.charAt(j)) {
               dp[i][j] = dp[i + 1][j - 1] + 2;
            } else {
               dp[i][j] = Math.max(dp[i + 1][j], dp[i][j - 1]);
            }
         }
      }
      return dp[0][n - 1];
   }
}
```

Output:

Length of Longest Palindromic Subsequence: 4

54. Implement the Rabin-Karp Algorithm for Pattern Matching

```
public class RabinKarpAlgorithm {
   public static void main(String[] args) {
      String text = "GEEKS FOR GEEKS";
      String pattern = "GEEK";
      int index = rabinKarp(text, pattern);
      System.out.println("Pattern found at index: " + index);
   }
public static int rabinKarp(String text, String pattern) {
      int m = pattern.length();
      int n = text.length();
      int q = 101; // A prime number
      int d = 256; // Number of characters in input alphabet
      int p = 0; // Hash value for pattern
      int t = 0; // Hash value for text
      int h = 1;
for (int i = 0; i < m - 1; i++) {
         h = (h * d) % q;
      }
for (int i = 0; i < m; i++) {
         p = (d * p + pattern.charAt(i)) % q;
         t = (d * t + text.charAt(i)) % q;
      }
for (int i = 0; i <= n - m; i++) {
         if (p == t) {
            boolean flag = true;
            for (int j = 0; j < m; j++) {
               if (text.charAt(i + j) != pattern.charAt(j)) {
                  flag = false;
                  break;
               }
            }
            if (flag) return i;
         }
         if (i < n - m) {
            t = (d * (t - text.charAt(i) * h) + text.charAt(i + 1)) % q;
```

```
start++;
end--;
}
if(isPalindrome){
System.out.println(str+"isapalindrome.");
}else{
System.out.println(str+"isnotapalindrome.");
}
}
}
```

//Palindrome

```
publicclassPalendrom{
publicstaticvoidmain(String[]args){
Stringst="madam";
Stringrev="";
for(inti=st.length()-1;i>=0;i--){
rev=rev+st.charAt(i);
}
System.out.println("ReverseName:"+rev);
if(st.equals(rev)){
System.out.println("palindrome");
}
else{
System.out.println("Notpalindrome");
}
}
}
```

Sort Name all way -

```
publicclasssort_long_Name{
publicstaticvoidmain(String[]args){
Stringstr2="Rishisinghkumarjaintyagi";
String[]words=str2.split("");
//System.out.println(words.length);
StringBuilderresult=newStringBuilder();
for(inti=0;i<words.length;i++){
if(i<words.length-1){
result.append(words[i].charAt(0)).append("");
}else{
result.append(words[i]);
}
}
System.out.println(result);
}
}  //output - R s k j tyagi

publicclassSortLongName2{
publicstaticvoidmain(String[]args){
Scannerin=newScanner(System.in);
System.out.println("Enteryourname:");
Stringstr=in.nextLine();
inti,j=0;
charc,d;

i=str.length()-1;
while(str.charAt(i)!=''){
i--;
}
while(j<i){
if(j==0){
System.out.print(str.charAt(j)+"");
}else{
c=str.charAt(j);
d=str.charAt(i+1);
if (flag) return i;
}
if (i < n - m) {
t = (d * (t - text.charAt(i) * h) + text.charAt(i + 1)) % q;
if (t < 0) t = t + q;
}
}
return -1;
}
}
```

Output:

Pattern found at index: 0

55. Check if a String is a Subsequence of Another String

```
public class CheckSubsequence {
  public static void main(String[] args) {
    String s1 = "abc";
    String s2 = "ahbgdc";
    boolean isSubsequence = isSubsequence(s1, s2);
    System.out.println("Is Subsequence: " + isSubsequence);
  }
public static boolean isSubsequence(String s1, String s2) {
    int i = 0, j = 0;
    while (i < s1.length() && j < s2.length()) {
      if (s1.charAt(i) == s2.charAt(j)) {
        i++;
      }
      j++;
    }
    return i == s1.length();
  }
}
```

Output:

Is Subsequence: true

Chapter 6: Advanced String Manipulations

56. Count Number of Substrings with Exactly K Distinct Characters

```
import java.util.HashMap;
import java.util.Map;
public class SubstringsWithKDistinctChars {
  public static void main(String[] args) {
    String str = "pqpqs";
    int k = 2;
    int count = countSubstringsWithKDistinctChars(str, k);
    System.out.println("Number of Substrings with Exactly " + k + "
Distinct Characters: " + count);
  }
public static int countSubstringsWithKDistinctChars(String s, int k) {
    int n = s.length();
    int res = 0;
    Map<Character, Integer> map = new HashMap<>();
    int left = 0;
    for (int right = 0; right < n; right++) {
      map.put(s.charAt(right), map.getOrDefault(s.charAt(right), 0) + 1);
      while (map.size() > k) {
        map.put(s.charAt(left), map.get(s.charAt(left)) - 1);
        if (map.get(s.charAt(left)) == 0) {
          map.remove(s.charAt(left));
        }
        left++;
      }
      res += right - left + 1;
    }
    return res;
  }
}
```

```
System.out.print(str.charAt(j)+" ");
}else{
c=str.charAt(j);
d=str.charAt(j+1);
if(c==''&&d!=''){
System.out.print(d+"");
}
}
j++;
}
System.out.print(str.substring(i));
}
}  //output -
Enter your name:
Rishi singh Rajput
R s  Rajput
```

Largest String find -

```
publicclasslargestString{
//publicstaticStringlargestStirng1(Stringstr,Stringstr2,Stringstr3){
publicstaticStringlargestString1(Stringstr,intsi,intei){
Stringsubstr="";
for(inti=si;i<ei;i++){
substr+=str.charAt(i);
}
returnsubstr;
//if(str.length()>str2.length()&&str.length()>str3.length()){
//System.out.println(str.toString()+""+"biggest1");
//}
//elseif(str2.length()>str3.length())
//{
//System.out.println(str2.toString()+""+"biggest2");
//}else{
//System.out.println(str3.toString()+""+"biggest3");
//}
//returnstr;
}
publicstaticvoidmain(String[]args){//o(n)timecomplex
//System.out.println(largestStirng1("Rishisingh","Monika","Gyani"
));
//largestString1("Rishisingh","Monika","Gyani");
Stringfruits[]={"apple","mango","banana"};
Stringlargest=fruits[0];
for(inti=0;i<fruits.length;i++){
if(largest.compareTo(fruits[i])<0){
largest=fruits[i];
}
}
System.out.println(largest);
}
}
```

All Naming Case -

```
publicstaticStringtoCamelCase(Stringstr){
String[]words=str.split("[\\s_\\-]+");
StringBuildercamelCaseString=newStringBuilder();
for(inti=0;i<words.length;i++){
Stringword=words[i];
if(i==0){
camelCaseString.append(word.toLowerCase());
}else{
camelCaseString.append(Character.toUpperCase(word.charAt(0)))
.append(word.substring(1).toLowerCase());
}
}
```

```
    return res;
  }
}
```

Output:

Number of Substrings with Exactly 2 Distinct Characters: 7

57. Find the Smallest Window in a String Containing All Characters of Another String

```
import java.util.HashMap;
import java.util.Map;
public class SmallestWindowContainingAllChars {
  public static void main(String[] args) {
    String str = "ADOBECODEBANC";
    String pat = "ABC";
    String result = findSmallestWindow(str, pat);
    System.out.println("Smallest Window: " + result);
  }
public static String findSmallestWindow(String str, String pat) {
    if (str.length() < pat.length()) return "";
    Map<Character, Integer> patMap = new HashMap<>();
    for (char ch : pat.toCharArray()) {
      patMap.put(ch, patMap.getOrDefault(ch, 0) + 1);
    }
int start = 0, startIndex = -1, minLen = Integer.MAX_VALUE;
    int count = 0;
    Map<Character, Integer> strMap = new HashMap<>();
    for (int j = 0; j < str.length(); j++) {
      char ch = str.charAt(j);
      strMap.put(ch, strMap.getOrDefault(ch, 0) + 1);
if (patMap.containsKey(ch) && strMap.get(ch) <= patMap.get(ch)) {
        count++;
      }
if (count == pat.length()) {
        while (!patMap.containsKey(str.charAt(start)) ||
strMap.get(str.charAt(start)) > patMap.get(str.charAt(start))) {
          if (strMap.get(str.charAt(start)) >
patMap.get(str.charAt(start))) {
            strMap.put(str.charAt(start), strMap.get(str.charAt(start)) -
1);
          }
          start++;
        }
int windowLen = j - start + 1;
        if (minLen > windowLen) {
          minLen = windowLen;
          startIndex = start;
        }
      }
    }
if (startIndex == -1) {
      return "";
    }
    return str.substring(startIndex, startIndex + minLen);
  }
}
```

Output:

Smallest Window: BANC

58. Longest Substring Without Repeating Characters

```
import java.util.HashSet;
```

Chapter 7: Advanced String Manipulations (Continued)

61. Count Occurrences of a Substring in a String

```
public class CountSubstringOccurrences {
  public static void main(String[] args) {
```

```
camelCaseString.append(Character.toUpperCase(word.charAt(0)))
.append(word.substring(1).toLowerCase());
}
}
returncamelCaseString.toString();
}

publicstaticStringtoPascalCase(Stringstr){
String[]words=str.split("[\\s_\\-]+");
StringBuilderpascalCaseString=newStringBuilder();
for(Stringword:words){
pascalCaseString.append(Character.toUpperCase(word.charAt(0))
)
.append(word.substring(1).toLowerCase());
}
returnpascalCaseString.toString();
}

publicstaticStringtoSnakeCase(Stringstr){
String[]words=str.split("[\\s_\\-]+");
StringBuildersnakeCaseString=newStringBuilder();
for(inti=0;i<words.length;i++){
if(i>0){
snakeCaseString.append("_");
}
snakeCaseString.append(words[i].toLowerCase());
}
returnsnakeCaseString.toString();
}

publicstaticStringtoKebabCase(Stringstr){
String[]words=str.split("[\\s_\\-]+");
StringBuilderkebabCaseString=newStringBuilder();
for(inti=0;i<words.length;i++){
if(i>0){
kebabCaseString.append("-");
}
kebabCaseString.append(words[i].toLowerCase());
}
returnkebabCaseString.toString();
}

publicstaticvoidmain(String[]args){
StringoriginalString="helloworld_thisisJava";

StringcamelCase=toCamelCase(originalString);
StringpascalCase=toPascalCase(originalString);
StringsnakeCase=toSnakeCase(originalString);
StringkebabCase=toKebabCase(originalString);

System.out.println("Original:"+originalString);
System.out.println("CamelCase:"+camelCase);
System.out.println("PascalCase:"+pascalCase);
System.out.println("SnakeCase:"+snakeCase);
System.out.println("KebabCase:"+kebabCase);
}
}
}
```

Output -

3. Find the Length of a String

```
java
public class StringLength {
   public static void main(String[] args) {
      String str = "Hello";
      int length = str.length();
```

61. Count Occurrences of a Substring in a String

```
public class CountSubstringOccurrences {
   public static void main(String[] args) {
      String str = "ABABABA";
      String subStr = "ABA";
      int count = countOccurrences(str, subStr);
      System.out.println("Occurrences of '" + subStr + "': " + count);
   }
public static int countOccurrences(String str, String subStr) {
      int count = 0;
      int index = 0;
      while ((index = str.indexOf(subStr, index)) != -1) {
         count++;
         index += subStr.length();
      }
      return count;
   }
}
```

Output:

```
Occurrences of 'ABA': 4
```

62. Check if Two Strings are Anagrams

```
import java.util.Arrays;
public class CheckAnagrams {
   public static void main(String[] args) {
      String str1 = "listen";
      String str2 = "silent";
      boolean isAnagram = areAnagrams(str1, str2);
      System.out.println("Are Anagrams: " + isAnagram);
   }
public static boolean areAnagrams(String str1, String str2) {
      char[] arr1 = str1.toCharArray();
      char[] arr2 = str2.toCharArray();
      Arrays.sort(arr1);
      Arrays.sort(arr2);
      return Arrays.equals(arr1, arr2);
   }
}
```

Output:

```
Are Anagrams: true
```

63. Reverse Words in a String

```
public class ReverseWords {
   public static void main(String[] args) {
      String str = "Hello World";
      String reversed = reverseWords(str);
      System.out.println("Reversed Words: " + reversed);
   }
public static String reverseWords(String s) {
      String[] words = s.split("\\s+");
      StringBuilder reversed = new StringBuilder();
      for (int i = words.length - 1; i >= 0; i--) {
         reversed.append(words[i]);
         if (i > 0) reversed.append(" ");
      }
      return reversed.toString();
   }
}
```

Output:

```
Reversed Words: World Hello
```

3.leetCode.ReverseName;

```
importjava.util.Scanner;
publicclassReverseName{
//Methodtoreverseastring
publicstaticStringreverse(Stringstr){
```

```
public class StringLength {
    public static void main(String[] args) {
        String str = "Hello";
        int length = str.length();
        System.out.println("Length of String: " + length);
    }
}
```

Output:

Length of String: 5

4. Convert String to Uppercase

java

```
public class StringUppercase {
    public static void main(String[] args) {
        String str = "hello";
        String uppercase = str.toUpperCase();
        System.out.println("Uppercase String: " + uppercase);
    }
}
```

Output:

Uppercase String: HELLO

5. Convert String to Lowercase

java

```
public class StringLowercase {
    public static void main(String[] args) {
        String str = "HELLO";
        String lowercase = str.toLowerCase();
        System.out.println("Lowercase String: " + lowercase);
    }
}
```

Output:

Lowercase String: hello

6. Concatenate Two Strings

```
public class StringConcatenate {
    public static void main(String[] args) {
        String str1 = "Hello";
        String str2 = "World";
        String concatenated = str1 + " " + str2;
        System.out.println("Concatenated String: " + concatenated);
    }
}
```

Output

Concatenated String: Hello World

7. Check if String Contains a Substring

```
public class StringContains {
    public static void main(String[] args) {
        String str = "Hello World";
        boolean contains = str.contains("World");
        System.out.println("Contains 'World': " + contains);
    }
}
```

Output:

Contains 'World': true

8. Find the Index of a Substring

```
public class SubstringIndex {
    public static void main(String[] args) {
        String str = "Hello World";
        int index = str.indexOf("World");
        System.out.println("Index of 'World': " + index);
    }
}
```

Output:

Index of 'World': 6

9. Extract a Substring

```
public class ExtractSubstring {
    public static void main(String[] args) {
```

```
publicclassReverseName{
//Methodtoreverseastring
publicstaticStringreverse(Stringstr){
//Convertthestringtoacharacterarray
char[]st=str.toCharArray();
intstart=0;
intend=st.length-1;
chartemp;
//Looptoswapcharactersfromthestartandenduntilthemiddleisreached
while(start<end){
//Swapcharacters
temp=st[start];
st[start]=st[end];
st[end]=temp;
start++;
end--;
}
//Convertthecharacterarraybacktoastringandreturnit
returnnewString(st);
}
publicstaticvoidmain(String[]args){
System.out.println("Enteryourname:");
Scannerscanner=newScanner(System.in);
//Readtheinputfromtheuser
Stringinput=scanner.nextLine();
//Callthereversemethodwiththeinputstring
StringreversedName=reverse(input);
//Printthereversedname
System.out.println("Yourreversednameis:"+reversedName);
}
}
//output
Enter your name:
4525245
Your reversed name is: 5425254
```

3.leetCode.ReverseName;

```
publicclassReverseName_Deepaksir{
publicstaticvoidmain(String[]args){
Stringst="RishisinghisaHero";
Stringrev="";
for(inti=st.length()-1;i>=0;i--){
rev=rev+st.charAt(i);
}
System.out.println("ReverseName:"+rev);
//IwanttoprintHeroaissinghRishi
Strings[]="RishisinghisaHero".split("");
Stringans="";
for(inti=s.length-1;i>=0;i--){
ans=ans+s[i]+"";
}
//System.out.println(ans.substring(0,ans.length()-1));
System.out.println(ans);
}
}
//output -
Reverse Name : oreH a si hgnis ihsiR
Hero a is singh Rishi
```

3.leetCode.ReverseName;

```
publicclassReverseName_use_charAt{
staticStringreverseName(Stringst){
char[]reversed=newchar[st.length()];
for(inti=0;i<st.length();i++){
```

Index of 'World': 6

9. Extract a Substring

```
public class ExtractSubstring {
    public static void main(String[] args) {
        String str = "Hello World";
        String substring = str.substring(6);
        System.out.println("Extracted Substring: " + substring);
    }
}
```

Output:

Extracted Substring: World

10. Replace Characters in a String

```
public class ReplaceCharacters {
    public static void main(String[] args) {
        String str = "Hello World";
        String replaced = str.replace('o', 'a');
        System.out.println("Replaced String: " + replaced);
    }
}
```

Output:

Replaced String: Hella Warld

Chapter 2: Intermediate String Operations

11. Split a String

```
import java.util.Arrays;
public class SplitString {
    public static void main(String[] args) {
        String str = "Hello World";
        String[] parts = str.split(" ");
        System.out.println("Split String: " + Arrays.toString(parts));
    }
}
```

Output:

Split String: [Hello, World]

12. Join an Array of Strings

```
public class JoinStrings {
    public static void main(String[] args) {
        String[] parts = {"Hello", "World"};
        String joined = String.join(" ", parts);
        System.out.println("Joined String: " + joined);
    }
}
```

Output:

Joined String: Hello World

13. Check if String Starts with a Prefix

```
public class StringStartsWith {
    public static void main(String[] args) {
        String str = "Hello World";
        boolean startsWith = str.startsWith("Hello");
        System.out.println("Starts with 'Hello': " + startsWith);
    }
}
```

Output:

Starts with 'Hello': true

14. Check if String Ends with a Suffix

java

```
public class StringEndsWith {
    public static void main(String[] args) {
        String str = "Hello World";
        boolean endsWith = str.endsWith("World");
        System.out.println("Ends with 'World': " + endsWith);
    }
}
```

Output:

```
staticStringreverseName(Stringst){
char[]reversed=newchar[st.length()];
for(inti=0;i<st.length();i++){
reversed[i]=st.charAt(st.length()-1-i);
}
returnnewString(reversed);
}
publicstaticvoidmain(String[]args){
Stringname="Rishi";
StringreversedName=reverseName(name);
System.out.println(STR."ReversedName:\{reversedName}");
}
}
/*Sure,let'sdryrunthecodestepbystep:
1.**Input**:`name="Rishi"`
2.The`reverseName`methodiscalledwiththeinputstring`"Rishi"`.
3.Insidethe`reverseName`method:
Acharacterarray`reversed`iscreatedwiththesamelengthastheinputstring,
whichis5inthiscase.
-Aloopiteratesfrom`i=0`to`i=4`.
-Ineachiteration:
-`reversed[i]`isassignedthecharacterattheindex`(length-1-
i)`oftheinputstring`"Rishi"`.So:
-When`i=0`,`reversed[0]=st.charAt(5-1-0)=st.charAt(4)='i'`
-When`i=1`,`reversed[1]=st.charAt(5-1-1)=st.charAt(3)='s'`
-When`i=2`,`reversed[2]=st.charAt(5-1-2)=st.charAt(2)='h'`
-When`i=3`,`reversed[3]=st.charAt(5-1-3)=st.charAt(1)='i'`
-When`i=4`,`reversed[4]=st.charAt(5-1-4)=st.charAt(0)='R'`
4.Aftertheloop,the`reversed`arraycontainsthecharacters`{'i','s','h','i','R'}`.
5.The`newString(reversed)`createsanewstringfromthecharacterarray,resu
ltinginthestring`"ishiR"`.
6.**Output**:`"ishiR"`
So,thereversednameof`"Rishi"`is`"ishiR"`.
*/
```

3.leetCode.ReverseName;

```
publicclassReverseName_use_toCharArray{
staticStringreverseName(Stringst){
char[]chars=st.toCharArray();
intstart=0;
intend=chars.length-1;
while(start<end){
chartemp=chars[start];
chars[start]=chars[end];
chars[end]=temp;
start++;
end--;
}
returnnewString(chars);
}
publicstaticvoidmain(String[]args){
Stringinput="Rishisingh";
Stringresult=reverseName(input);
System.out.println("ReversedName:"+result);
}
}
```

//output -

Reversed Name: hgnis ihsiR

```
publicstaticvoidmain(String[]args){
Stringtext="IamHero";
Stringstr[]=text.split("");
Collections.reverse(Arrays.asList(str));
System.out.println(String.join("",str));
```

```
    }
}
```
Output:
Ends with 'World': true

15. Count Occurrences of a Character
```
public class CountCharacterOccurrences {
    public static void main(String[] args) {
        String str = "Hello World";
        char ch = 'o';
        int count = 0;
        for (int i = 0; i < str.length(); i++) {
            if (str.charAt(i) == ch) {
                count++;
            }
        }
        System.out.println("Occurrences of 'o': " + count);
    }
}
```
Output:
Occurrences of 'o': 2

16. Remove Whitespace from a String
```
public class RemoveWhitespace {
    public static void main(String[] args) {
        String str = "  Hello   World  ";
        String trimmed = str.trim();
        System.out.println("String without leading and trailing
whitespace: '" + trimmed + "'");
    }
}
```
Output:
String without leading and trailing whitespace: 'Hello   World'

17. Check if String is Empty
```
public class CheckEmptyString {
    public static void main(String[] args) {
        String str = "";
        boolean isEmpty = str.isEmpty();
        System.out.println("Is String Empty: " + isEmpty);
    }
}
```
Output:
Is String Empty: true

18. Convert String to Character Array
```
import java.util.Arrays;
public class StringToCharArray {
    public static void main(String[] args) {
        String str = "Hello";
        char[] charArray = str.toCharArray();
        System.out.println("Character Array: " +
Arrays.toString(charArray));
    }
}
```
Output:
Character Array: [H, e, l, l, o]

19. Convert Character Array to String
```
public class CharArrayToString {
    public static void main(String[] args) {
        char[] charArray = {'H', 'e', 'l', 'l', 'o'};
        String str = new String(charArray);
        System.out.println("String: " + str);
    }
}
```
Output:
String: Hello

20. Compare Two Strings
```
public class CompareStrings {
Stringstr[]=text.split("");
Collections.reverse(Arrays.asList(str));
System.out.println(String.join("",str));
}
}  //output - Hero am I
```

64. Find All Permutations of a String
```
import java.util.HashSet;
import java.util.Set;
public class Permutations {
    public static void main(String[] args) {
        String str = "ABC";
        Set<String> permutations = new HashSet<>();
        generatePermutations("", str, permutations);
        System.out.println("Permutations: " + permutations);
    }
public static void generatePermutations(String prefix, String str,
Set<String> result) {
        int n = str.length();
        if (n == 0) {
            result.add(prefix);
        } else {
            for (int i = 0; i < n; i++) {
                generatePermutations(prefix + str.charAt(i), str.substring(0, i) +
str.substring(i + 1, n), result);
            }
        }
    }
}
```
Output:
makefile
Copy code
Permutations: [ACB, CAB, CBA, BAC, ABC, BCA]

65. Remove Duplicates from a String
```
public class RemoveDuplicates {
    public static void main(String[] args) {
        String str = "aabbcc";
        String result = removeDuplicates(str);
        System.out.println("String after removing duplicates: " + result);
    }
public static String removeDuplicates(String s) {
        StringBuilder result = new StringBuilder();
        boolean[] seen = new boolean[256];
        for (char c : s.toCharArray()) {
            if (!seen[c]) {
                seen[c] = true;
                result.append(c);
            }
        }
        return result.toString();
    }
}
```
Output:
String after removing duplicates: abc

66. Check if String is a Rotation of Another String
```
public class CheckRotation {
    public static void main(String[] args) {
        String str1 = "waterbottle";
        String str2 = "erbottlewat";
        boolean isRotation = isRotation(str1, str2);
        System.out.println("Is Rotation: " + isRotation);
    }
public static boolean isRotation(String str1, String str2) {
        if (str1.length() != str2.length()) return false;
```

Output:
String: Hello

20. Compare Two Strings

```
public class CompareStrings {
   public static void main(String[] args) {
      String str1 = "Hello";
      String str2 = "World";
      int result = str1.compareTo(str2);
      System.out.println("Comparison Result: " + result);
   }
}
```

Output:
Comparison Result: -15

Chapter 3: Advanced String Operations

21. Find All Permutations of a String

```
public class StringPermutations {
   public static void main(String[] args) {
      String str = "ABC";
      permute(str, "");
   }
public static void permute(String str, String result) {
      if (str.length() == 0) {
         System.out.println(result);
         return;
      }
for (int i = 0; i < str.length(); i++) {
         char ch = str.charAt(i);
         String ros = str.substring(0, i) + str.substring(i + 1);
         permute(ros, result + ch);
      }
   }
}
```

Output:
ABC
ACB
BAC
BCA
CAB
CBA

22. Remove Duplicates from a String

```
public class RemoveDuplicates {
   public static void main(String[] args) {
      String str = "hello";
      String result = removeDuplicates(str);
      System.out.println("String without duplicates: " + result);
   }
public static String removeDuplicates(String str) {
      Set<Character> seen = new HashSet<>();
      StringBuilder sb = new StringBuilder();
      for (char ch : str.toCharArray()) {
         if (!seen.contains(ch)) {
            seen.add(ch);
            sb.append(ch);
         }
      }
      return sb.toString();
   }
}
```

Output:
String without duplicates: helo

23. Check if Two Strings are Anagrams

```
public class AnagramCheck {
   public static void main(String[] args) {
   }
public static boolean isRotation(String str1, String str2) {
      if (str1.length() != str2.length()) return false;
      String combined = str1 + str1;
      return combined.contains(str2);
   }
}
```

Output:
Is Rotation: true

67. Find the First Non-Repeating Character

```
import java.util.LinkedHashMap;
import java.util.Map;
public class FirstNonRepeatingCharacter {
   public static void main(String[] args) {
      String str = "swiss";
      char result = firstNonRepeatingCharacter(str);
      System.out.println("First Non-Repeating Character: " + result);
   }
public static char firstNonRepeatingCharacter(String s) {
      Map<Character, Integer> counts = new LinkedHashMap<>();
      for (char c : s.toCharArray()) {
         counts.put(c, counts.getOrDefault(c, 0) + 1);
      }
      for (Map.Entry<Character, Integer> entry : counts.entrySet()) {
         if (entry.getValue() == 1) {
            return entry.getKey();
         }
      }
      return '\0'; // No non-repeating character
   }
}
```

Output:
First Non-Repeating Character: w

68. Count Vowels and Consonants in a String

```
public class CountVowelsConsonants {
   public static void main(String[] args) {
      String str = "Hello World";
      int[] count = countVowelsConsonants(str);
      System.out.println("Vowels: " + count[0] + ", Consonants: " +
count[1]);
   }
public static int[] countVowelsConsonants(String s) {
      int[] counts = new int[2]; // counts[0] for vowels, counts[1] for
consonants
      String vowels = "aeiouAEIOU";
      for (char c : s.toCharArray()) {
         if (Character.isLetter(c)) {
            if (vowels.indexOf(c) != -1) {
               counts[0]++;
            } else {
               counts[1]++;
            }
         }
      }
      return counts;
   }
}
```

Output:
Vowels: 3, Consonants: 7

69. Check if a String is a Valid Number

```
public class ValidNumber {
   public static void main(String[] args) {
      String str = "123.45";
      boolean isValid = isValidNumber(str);
```

23. Check if Two Strings are Anagrams

```
public class AnagramCheck {
   public static void main(String[] args) {
      String str1 = "listen";
      String str2 = "silent";
      boolean areAnagrams = areAnagrams(str1, str2);
      System.out.println("Are Anagrams: " + areAnagrams);
   }
public static boolean areAnagrams(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);
   }
}
```

Output:

Are Anagrams: true

24. Find the Longest Palindromic Substring

```
public class LongestPalindromicSubstring {
   public static void main(String[] args) {
      String str = "babad";
      String result = longestPalindrome(str);
      System.out.println("Longest Palindromic Substring: " +
result);
   }
public static String longestPalindrome(String s) {
      if (s == null || s.length() < 1) return "";
      int start = 0, end = 0;
      for (int i = 0; i < s.length(); i++) {
         int len1 = expandAroundCenter(s, i, i);
         int len2 = expandAroundCenter(s, i, i + 1);
         int len = Math.max(len1, len2);
         if (len > end - start) {
            start = i - (len - 1) / 2;
            end = i + len / 2;
         }
      }
      return s.substring(start, end + 1);
   }
private static int expandAroundCenter(String s, int left, int right) {
      int L = left, R = right;
      while (L >= 0 && R < s.length() && s.charAt(L) == s.charAt(R)) {
         L--;
         R++;
      }
      return R - L - 1;
   }
}
```

Output:

Longest Palindromic Substring: bab

25. Count Vowels and Consonants in a String

```
public class VowelsConsonantsCount {
   public static void main(String[] args) {
      String str = "Hello World";
      int vowels = 0, consonants = 0;
      str = str.toLowerCase();
      for (char ch : str.toCharArray()) {
         if (ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u') {
            vowels++;
         } else if (ch >= 'a' && ch <= 'z') {
   public static void main(String[] args) {
      String str = "123.45";
      boolean isValid = isValidNumber(str);
      System.out.println("Is Valid Number: " + isValid);
   }
public static boolean isValidNumber(String s) {
      try {
         Double.parseDouble(s);
         return true;
      } catch (NumberFormatException e) {
         return false;
      }
   }
}
```

Output:

Is Valid Number: true

70. Convert a String to a Number

```
public class StringToNumber {
   public static void main(String[] args) {
      String str = "12345";
      int number = stringToNumber(str);
      System.out.println("Converted Number: " + number);
   }
public static int stringToNumber(String s) {
      return Integer.parseInt(s);
   }
}
```

Output:

Converted Number: 12345

Chapter 8: String Comparison and Manipulation

71. Compare Two Strings Lexicographically

```
public class CompareStrings {
   public static void main(String[] args) {
      String str1 = "apple";
      String str2 = "banana";
      int comparison = str1.compareTo(str2);
      System.out.println("Comparison Result: " + comparison);
   }
}
```

Output:

Comparison Result: -1

72. Split a String into Tokens

```
public class SplitString {
   public static void main(String[] args) {
      String str = "Java,Python,C++";
      String[] tokens = str.split(",");
      System.out.println("Tokens: " + Arrays.toString(tokens));
   }
}
```

Output:

Tokens: [Java, Python, C++]

73. Concatenate Two Strings

```
public class ConcatenateStrings {
   public static void main(String[] args) {
      String str1 = "Hello ";
      String str2 = "World";
      String result = str1.concat(str2);
      System.out.println("Concatenated String: " + result);
   }
}
```

Output:

Concatenated String: Hello World

74. Repeat a String N Times

```
        if (ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u') {
            vowels++;
        } else if (ch >= 'a' && ch <= 'z') {
            consonants++;
        }
    }
    System.out.println("Vowels: " + vowels + ", Consonants: " +
consonants);
  }
}
```

Output:

Vowels: 3, Consonants: 7

26. Find the First Non-Repeated Character in a String

```
import java.util.LinkedHashMap;
import java.util.Map;
public class FirstNonRepeatedCharacter {
  public static void main(String[] args) {
    String str = "swiss";
    char result = firstNonRepeatedCharacter(str);
    System.out.println("First Non-Repeated Character: " + result);
  }
public static char firstNonRepeatedCharacter(String str) {
    Map<Character, Integer> charCountMap = new
LinkedHashMap<>();
    for (char ch : str.toCharArray()) {
      charCountMap.put(ch, charCountMap.getOrDefault(ch, 0)
+ 1);
    }
    for (Map.Entry<Character, Integer> entry :
charCountMap.entrySet()) {
      if (entry.getValue() == 1) {
        return entry.getKey();
      }
    }
    return '\0';
  }
}
```

Output:

First Non-Repeated Character: w

27. Check if a String is a Rotation of Another String

```
public class StringRotationCheck {
  public static void main(String[] args) {
    String str1 = "abcd";
    String str2 = "cdab";
    boolean isRotation = isRotation(str1, str2);
    System.out.println("Is Rotation: " + isRotation);
  }
public static boolean isRotation(String str1, String str2) {
    if (str1.length() != str2.length()) {
      return false;
    }
    String concatenated = str1 + str1;
    return concatenated.contains(str2);
  }
}
```

Output:

Is Rotation: true

28. Convert a String to an Integer

```
public class StringToInteger {
  public static void main(String[] args) {
    String str = "123";
    int num = Integer.parseInt(str);
    System.out.println("Integer: " + num);
  }
```

Output:

Concatenated String: Hello World

74. Repeat a String N Times

```
public class RepeatString {
  public static void main(String[] args) {
    String str = "Hi ";
    int n = 3;
    String result = str.repeat(n);
    System.out.println("Repeated String: " + result);
  }
}
```

Output:

Repeated String: Hi Hi Hi

76. Trim Whitespace from a String

```
public class TrimWhitespace {
  public static void main(String[] args) {
    String str = "  Hello World  ";
    String trimmed = str.trim();
    System.out.println("Trimmed String: '" + trimmed + "'");
  }
}
```

Output:

Trimmed String: 'Hello World'

77. Replace All Occurrences of a Substring

```
public class ReplaceSubstring {
  public static void main(String[] args) {
    String str = "hello world";
    String replaced = str.replace("world", "Java");
    System.out.println("Replaced String: " + replaced);
  }
}
```

Output:

Replaced String: hello Java

78. Remove All Occurrences of a Character

```
public class RemoveCharacter {
  public static void main(String[] args) {
    String str = "banana";
    char ch = 'a';
    String result = str.replace(String.valueOf(ch), "");
    System.out.println("String after removing '" + ch + "': " + result);
  }
}
```

Output:

String after removing 'a': bnn

79. Find the Last Occurrence of a Character

```
public class LastOccurrence {
  public static void main(String[] args) {
    String str = "hello world";
    char ch = 'o';
    int index = str.lastIndexOf(ch);
    System.out.println("Last occurrence of '" + ch + "': " + index);
  }
}
```

Output:

Last occurrence of 'o': 7

80. Find the Index of a Substring

```
public class IndexOfSubstring {
  public static void main(String[] args) {
    String str = "Java Programming";
    String subStr = "Programming";
    int index = str.indexOf(subStr);
    System.out.println("Index of '" + subStr + "': " + index);
  }
}
```

```
        int num = Integer.parseInt(str);
        System.out.println("Integer: " + num);
    }
}
```
Output:
Integer: 123

29. Convert an Integer to a String
```
public class IntegerToString {
    public static void main(String[] args) {
        int num = 123;
        String str = Integer.toString(num);
        System.out.println("String: " + str);
    }
}
```
Output:
String: 123

30. Check if a String is Numeric
```
public class CheckNumericString {
    public static void main(String[] args) {
        String str = "12345";
        boolean isNumeric = str.chars().allMatch(Character::isDigit);
        System.out.println("Is Numeric: " + isNumeric);
    }
}
```
Output:
Is Numeric: true

31. Find the Most Frequent Character in a String
```
public class MostFrequentCharacter {
    public static void main(String[] args) {
        String str = "success";
        char result = mostFrequentCharacter(str);
        System.out.println("Most Frequent Character: " + result);
    }
public static char mostFrequentCharacter(String str) {
 Map<Character, Integer> charCountMap = new HashMap<>();
  for (char ch : str.toCharArray()) {
    charCountMap.put(ch, charCountMap.getOrDefault(ch, 0) + 1);
        }
        char mostFrequent = '\0';
        int maxCount = 0;
        for (Map.Entry<Character, Integer> entry :
charCountMap.entrySet()) {
            if (entry.getValue() > maxCount) {
                maxCount = entry.getValue();
                mostFrequent = entry.getKey();
            }
        }
        return mostFrequent;
    }
}
```
Output:
Most Frequent Character: s

32. Count Words in a String
```
public class WordCount {
    public static void main(String[] args) {
        String str = "Hello World";
        String[] words = str.split("\\s+");
        int wordCount = words.length;
        System.out.println("Word Count: " + wordCount);
    }
}
```
Output:
Word Count: 2

34. Capitalize the First Letter of Each Word in a String
```
        System.out.println("Index of '" + subStr + "': " + index);
    }
}
```
Output:
Index of 'Programming': 5

81. Check if String Starts or Ends with a Substring
```
public class CheckStartEnd {
    public static void main(String[] args) {
        String str = "Java Programming";
        String start = "Java";
        String end = "Programming";
        boolean startsWith = str.startsWith(start);
        boolean endsWith = str.endsWith(end);
        System.out.println("Starts with '" + start + "': " + startsWith);
        System.out.println("Ends with '" + end + "': " + endsWith);
    }
}
```
Output:
Starts with 'Java': true
Ends with 'Programming': true

82. Convert String to Char Array
```
public class StringToCharArray {
    public static void main(String[] args) {
        String str = "Hello";
        char[] charArray = str.toCharArray();
        System.out.println("Char Array: " + Arrays.toString(charArray));
    }
}
```
Output:
Char Array: [H, e, l, l, o]

83. Convert Char Array to String
```
public class CharArrayToString {
    public static void main(String[] args) {
        char[] charArray = {'H', 'e', 'l', 'l', 'o'};
        String str = new String(charArray);
        System.out.println("String: " + str);
    }
}
```
Output:
String: Hello

84. Check if String Contains a Substring
```
public class ContainsSubstring {
    public static void main(String[] args) {
        String str = "hello world";
        String subStr = "world";
        boolean contains = str.contains(subStr);
        System.out.println("Contains '" + subStr + "': " + contains);
    }
}
```
Output:
Contains 'world': true

85. Find the Length of a String
```
public class StringLength {
    public static void main(String[] args) {
        String str = "Hello World";
        int length = str.length();
        System.out.println("Length of the string: " + length);
    }
}
```
Output:
Length of the string: 11

86. Convert String to Byte Array
```
import java.nio.charset.StandardCharsets;
public class StringToByteArray {
```

Output:
Word Count: 2

34. Capitalize the First Letter of Each Word in a String

```
public class CapitalizeWords {
   public static void main(String[] args) {
      String str = "hello world";
      String[] words = str.split("\\s+");
      StringBuilder capitalized = new StringBuilder();
      for (String word : words)
{ capitalized.append(Character.toUpperCase(word.charAt(0))).
append(word.substring(1)).append(" ");
      }
      System.out.println("Capitalized Words: " +
capitalized.toString().trim());
   }
}
```

Output:
Capitalized Words: Hello World

35. Check if a String is a Valid IP Address

```
import java.util.regex.Pattern;
public class ValidateIPAddress {
   public static void main(String[] args) {
      String ip = "192.168.0.1";
      boolean isValid = isValidIPAddress(ip);
      System.out.println("Is Valid IP Address: " + isValid);
   }
public static boolean isValidIPAddress(String ip) {
      String regex =
         "^(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\\." +
         "(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\\." +
         "(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\\." +
         "(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)$";
      return Pattern.matches(regex, ip);
   }
}
```

Output:
Is Valid IP Address: true

36. Remove a Given Character from a String

```
public class RemoveCharacter {
   public static void main(String[] args) {
      String str = "hello world";
      char ch = 'o';
      String result = str.replace(Character.toString(ch), "");
      System.out.println("String after removing '" + ch + "': " +
result);
   }
}
```

Output:
String after removing 'o': hell wrld

37. Convert a String to Title Case

```
public class TitleCaseString {
   public static void main(String[] args) {
      String str = "hello world";
      String[] words = str.split("\\s+");
      StringBuilder titleCase = new StringBuilder();
      for (String word : words) {
         titleCase.append(Character.toUpperCase(word.charAt(0)))
               .append(word.substring(1).toLowerCase())
               .append(" ");
      }
      System.out.println("Title Case String: " +
titleCase.toString().trim());
   }
}
```

86. Convert String to Byte Array

```
import java.nio.charset.StandardCharsets;
public class StringToByteArray {
   public static void main(String[] args) {
      String str = "Hello";
      byte[] byteArray = str.getBytes(StandardCharsets.UTF_8);
      System.out.println("Byte Array: " + Arrays.toString(byteArray));
   }
}
```

Output:
Byte Array: [72, 101, 108, 108, 111]

87. Convert Byte Array to String

```
import java.nio.charset.StandardCharsets;
public class ByteArrayToString {
   public static void main(String[] args) {
      byte[] byteArray = {72, 101, 108, 108, 111};
      String str = new String(byteArray, StandardCharsets.UTF_8);
      System.out.println("String: " + str);
   }
}
```

Output:
String: Hello

88. Check if a String is a Palindrome

```
public class CheckPalindrome {
   public static void main(String[] args) {
      String str = "racecar";
      boolean isPalindrome = isPalindrome(str);
      System.out.println("Is Palindrome: " + isPalindrome);
   }
public static boolean isPalindrome(String s) {
      int left = 0, right = s.length() - 1;
      while (left < right) {
         if (s.charAt(left) != s.charAt(right)) {
            return false;
         }
         left++;
         right--;
      }
      return true;
   }
}
```

Output:
Is Palindrome: true

89. Find the Most Frequent Character in a String

```
import java.util.HashMap;
import java.util.Map;
public class MostFrequentCharacter {
   public static void main(String[] args) {
      String str = "abacabad";
      char result = mostFrequentCharacter(str);
      System.out.println("Most Frequent Character: " + result);
   }
public static char mostFrequentCharacter(String s) {
      Map<Character, Integer> counts = new HashMap<>();
      for (char c : s.toCharArray()) {
         counts.put(c, counts.getOrDefault(c, 0) + 1);
      }
      return counts.entrySet().stream()
            .max(Map.Entry.comparingByValue())
            .get()
            .getKey();
   }
}
```

Output:

```
titleCase.toString().trim());
    }
}
```
Output:
Title Case String: Hello World
38. Find the Longest Word in a String
```
public class LongestWord {
    public static void main(String[] args) {
        String str = "The quick brown fox jumped over the lazy dog";
        String[] words = str.split("\\s+");
        String longestWord = "";
        for (String word : words) {
            if (word.length() > longestWord.length()) {
                longestWord = word;
            }
        }
        System.out.println("Longest Word: " + longestWord);
    }
}
```
Output:
Longest Word: jumped
40. Find All Substrings of a String
Copy code
```
public class AllSubstrings {
    public static void main(String[] args) {
        String str = "abc";
        for (int i = 0; i < str.length(); i++) {
            for (int j = i + 1; j <= str.length(); j++) {
                System.out.println(str.substring(i, j));
            }
        }
    }
}
```
Output:
Copy code
a
ab
abc
b
bc
c

Chapter 4: Regular Expressions with Strings

41. Validate an Email Address
```
import java.util.regex.Pattern;
public class ValidateEmailAddress {
    public static void main(String[] args) {
        String email = "example@example.com";
        boolean isValid = isValidEmail(email);
        System.out.println("Is Valid Email Address: " + isValid);
    }
public static boolean isValidEmail(String email) {
        String regex = "^[A-Za-z0-9+_.-]+@(.+)$";
        return Pattern.matches(regex, email);
    }
}
```
Output:
Is Valid Email Address: true
42. Validate a Phone Number
```
import java.util.regex.Pattern;
public class ValidatePhoneNumber {
    public static void main(String[] args) {
        String phone = "123-456-7890";
        boolean isValid = isValidPhoneNumber(phone);
    }
}
```
Output:
Most Frequent Character: a
90. Replace First Occurrence of a Substring
```
public class ReplaceFirstOccurrence {
    public static void main(String[] args) {
        String str = "hello world hello";
        String result = str.replaceFirst("hello", "hi");
        System.out.println("String after replacement: " + result);
    }
}
```
Output:
String after replacement: hi world hello
91. Replace All Non-Alphanumeric Characters
```
public class ReplaceNonAlphanumeric {
    public static void main(String[] args) {
        String str = "Hello @ World!";
        String result = str.replaceAll("[^a-zA-Z0-9]", "");
        System.out.println("String after replacing non-alphanumeric
characters: " + result);
    }
}
```
Output:
String after replacing non-alphanumeric characters: HelloWorld
92. Find the Longest Common Prefix
```
public class LongestCommonPrefix {
    public static void main(String[] args) {
        String[] strs = {"flower", "flow", "flight"};
        String result = longestCommonPrefix(strs);
        System.out.println("Longest Common Prefix: " + result);
    }
public static String longestCommonPrefix(String[] strs) {
        if (strs.length == 0) return "";
        String prefix = strs[0];
        for (int i = 1; i < strs.length; i++) {
            while (strs[i].indexOf(prefix) != 0) {
                prefix = prefix.substring(0, prefix.length() - 1);
                if (prefix.isEmpty()) return "";
            }
        }
        return prefix;
    }
}
```
Output:
Longest Common Prefix: fl
93. Count the Number of Words in a String
```
public class CountWords {
    public static void main(String[] args) {
        String str = "This is a test string.";
        int count = countWords(str);
        System.out.println("Number of Words: " + count);
    }
public static int countWords(String s) {
        String[] words = s.trim().split("\\s+");
        return words.length;
    }
}
```
Output:
Number of Words: 5
94. Remove All Instances of a Character
```
public class RemoveCharacterInstances {
    public static void main(String[] args) {
        String str = "banana";
```

```
    public static void main(String[] args) {
        String phone = "123-456-7890";
        boolean isValid = isValidPhoneNumber(phone);
        System.out.println("Is Valid Phone Number: " + isValid);
    }
public static boolean isValidPhoneNumber(String phone) {
        String regex = "^(\\d{3}[- .]?){2}\\d{4}$";
        return Pattern.matches(regex, phone);
    }
}
```

Output:

Is Valid Phone Number: true

43. Extract Digits from a String

```
public class ExtractDigits {
    public static void main(String[] args) {
        String str = "abc123def456";
        String digits = str.replaceAll("[^\\d]", "");
        System.out.println("Extracted Digits: " + digits);
    }
}
```

Output:

Extracted Digits: 123456

44. Extract Letters from a String

```
public class ExtractLetters {
    public static void main(String[] args) {
        String str = "abc123def456";
        String letters = str.replaceAll("[^a-zA-Z]", "");
        System.out.println("Extracted Letters: " + letters);
    }
}
```

Output:

Extracted Letters: abcdef

45. Check if String Contains Only Letters

```
public class CheckLettersOnly {
    public static void main(String[] args) {
        String str = "HelloWorld";
        boolean isOnlyLetters = str.matches("[a-zA-Z]+");
        System.out.println("Contains Only Letters: " + isOnlyLetters);
    }
}
```

Output:

Contains Only Letters: true

46. Check if String Contains Only Digits

```
public class CheckDigitsOnly {
    public static void main(String[] args) {
        String str = "12345";
        boolean isOnlyDigits = str.matches("\\d+");
        System.out.println("Contains Only Digits: " + isOnlyDigits);
    }
}
```

Output:

Contains Only Digits: true

47. Extract Words from a String

```
import java.util.Arrays;
public class ExtractWords {
    public static void main(String[] args) {
        String str = "Hello, world! Welcome to Java.";
        String[] words = str.split("\\W+");
        System.out.println("Extracted Words: " +
Arrays.toString(words));
    }
}
```

Output:

Extracted Words: [Hello, world, Welcome, to, Java]

```
public class RemoveCharacterInstances {
    public static void main(String[] args) {
        String str = "banana";
        char ch = 'a';
        String result = str.replaceAll(Character.toString(ch), "");
        System.out.println("String after removing all instances of '" + ch + "':
" + result);
    }
}
```

Output:

String after removing all instances of 'a': bnn

95. Find Unique Characters in a String

```
import java.util.HashSet;
import java.util.Set;
public class UniqueCharacters {
    public static void main(String[] args) {
        String str = "aabbcc";
        Set<Character> uniqueChars = findUniqueCharacters(str);
        System.out.println("Unique Characters: " + uniqueChars);
    }
public static Set<Character> findUniqueCharacters(String s) {
        Set<Character> uniqueChars = new HashSet<>();
        for (char c : s.toCharArray()) {
            uniqueChars.add(c);
        }
        return uniqueChars;
    }
}
```

Output:

Unique Characters: [a, b, c]

96. Find Substrings of a Given Length

```
import java.util.HashSet;
import java.util.Set;
public class SubstringsOfLength {
    public static void main(String[] args) {
        String str = "abcabc";
        int length = 3;
        Set<String> substrings = findSubstrings(str, length);
        System.out.println("Substrings of length " + length + ": " +
substrings);
    }
public static Set<String> findSubstrings(String s, int length) {
        Set<String> substrings = new HashSet<>();
        for (int i = 0; i <= s.length() - length; i++) {
            substrings.add(s.substring(i, i + length));
        }
        return substrings;
    }
}
```

Output:

Substrings of length 3: [bca, abc, cab, cba]

97. Find the Shortest Unique Prefix for Each Word

```
import java.util.HashMap;
import java.util.Map;
public class ShortestUniquePrefix {
    public static void main(String[] args) {
        String[] words = {"zebra", "dog", "duck", "dove"};
        Map<String, String> result = findUniquePrefixes(words);
        System.out.println("Shortest Unique Prefixes: " + result);
    }
public static Map<String, String> findUniquePrefixes(String[] words) {
        Map<String, String> prefixes = new HashMap<>();
        for (String word : words) {
            String prefix = "";
```

```
}
```
Output:
```
Extracted Words: [Hello, world, Welcome, to, Java]
```
48. Replace All Digits with a Character
```
public class ReplaceDigits {
    public static void main(String[] args) {
        String str = "abc123def456";
        String replaced = str.replaceAll("\\d", "*");
        System.out.println("Replaced Digits: " + replaced);
    }
}
```
Output:
```
Replaced Digits: abc***def***
```
49. Validate a Date String
```
import java.util.regex.Pattern;
public class ValidateDate {
    public static void main(String[] args) {
        String date = "2021-12-31";
        boolean isValid = isValidDate(date);
        System.out.println("Is Valid Date: " + isValid);
    }
public static boolean isValidDate(String date) {
        String regex = "^\\d{4}-\\d{2}-\\d{2}$";
        return Pattern.matches(regex, date);
    }
}
```
Output:
```
Is Valid Date: true
```
50. Validate a Time String
```
import java.util.regex.Pattern;
public class ValidateTime {
    public static void main(String[] args) {
        String time = "23:59:59";
        boolean isValid = isValidTime(time);
        System.out.println("Is Valid Time: " + isValid);
    }
public static boolean isValidTime(String time) {
        String regex = "^([01]?[0-9]|2[0-3]):[0-5][0-9]:[0-5][0-9]$";
        return Pattern.matches(regex, time);
    }
}
```
Output:
```
Is Valid Time: true
```

Chapter 5: String Algorithms and Challenges

51. Find the Longest Common Prefix
```
public class LongestCommonPrefix {
    public static void main(String[] args) {
        String[] strs = {"flower", "flow", "flight"};
        String result = longestCommonPrefix(strs);
        System.out.println("Longest Common Prefix: " + result);
    }
public static String longestCommonPrefix(String[] strs) {
        if (strs == null || strs.length == 0) return "";
        String prefix = strs[0];
        for (int i = 1; i < strs.length; i++) {
            while (strs[i].indexOf(prefix) != 0) {
                prefix = prefix.substring(0, prefix.length() - 1);
                if (prefix.isEmpty()) return "";
            }
        }
        return prefix;
    }
}
```
```
        Map<String, String> prefixes = new HashMap<>();
        for (String word : words) {
            String prefix = "";
            for (int i = 0; i < word.length(); i++) {
                prefix += word.charAt(i);
                boolean isUnique = true;
                for (String other : words) {
                    if (!other.equals(word) && other.startsWith(prefix)) {
                        isUnique = false;
                        break;
                    }
                }
                if (isUnique) {
                    prefixes.put(word, prefix);
                    break;
                }
            }
        }
        return prefixes;
    }
}
```
Output:
```
Shortest Unique Prefixes: {zebra=z, dog=d, duck=du, dove=do}
```
98. Find All Palindromic Substrings
```
import java.util.HashSet;
import java.util.Set;
public class PalindromicSubstrings {
    public static void main(String[] args) {
        String str = "aba";
        Set<String> palindromes = findPalindromicSubstrings(str);
        System.out.println("Palindromic Substrings: " + palindromes);
    }
public static Set<String> findPalindromicSubstrings(String s) {
        Set<String> palindromes = new HashSet<>();
        for (int i = 0; i < s.length(); i++) {
            for (int j = i; j < s.length(); j++) {
                String substring = s.substring(i, j + 1);
                if (isPalindrome(substring)) {
                    palindromes.add(substring);
                }
            }
        }
        return palindromes;
    }
public static boolean isPalindrome(String s) {
        int left = 0, right = s.length() - 1;
        while (left < right) {
            if (s.charAt(left) != s.charAt(right)) {
                return false;
            }
            left++;
            right--;
        }
        return true;
    }
}
```
Output:
```
Palindromic Substrings: [a, aba, b, bab]
```
99. Count the Frequency of Each Character
```
import java.util.HashMap;
import java.util.Map;
public class CharacterFrequency {
    public static void main(String[] args) {
        String str = "programming";
```

```
        return prefix;
    }
}
```
Output:
Longest Common Prefix: fl

52. Implement the KMP Algorithm for Pattern Matching
```
public class KMPAlgorithm {
    public static void main(String[] args) {
        String text = "abxabcabcaby";
        String pattern = "abcaby";
        int index = KMPSearch(text, pattern);
        System.out.println("Pattern found at index: " + index);
    }
public static int KMPSearch(String text, String pattern) {
        int[] lps = computeLPSArray(pattern);
        int i = 0, j = 0;
        while (i < text.length()) {
            if (pattern.charAt(j) == text.charAt(i)) {
                i++;
                j++;
            }
            if (j == pattern.length()) {
                return i - j;
            } else if (i < text.length() && pattern.charAt(j) !=
text.charAt(i)) {
                if (j != 0) {
                    j = lps[j - 1];
                } else {
                    i++;
                }
            }
        }
        return -1;
    }
private static int[] computeLPSArray(String pattern) {
        int[] lps = new int[pattern.length()];
        int length = 0, i = 1;
        while (i < pattern.length()) {
            if (pattern.charAt(i) == pattern.charAt(length)) {
                length++;
                lps[i] = length;
                i++;
            } else {
                if (length != 0) {
                    length = lps[length - 1];
                } else {
                    lps[i] = length;
                    i++;
                }
            }
        }
        return lps;
    }
}
```
Output:
Pattern found at index: 6

Sort Name -

```
public class CharacterFrequency {
    public static void main(String[] args) {
        String str = "programming";
        Map<Character, Integer> frequency = countFrequency(str);
        System.out.println("Character Frequencies: " + frequency);
    }
public static Map<Character, Integer> countFrequency(String s) {
        Map<Character, Integer> frequency = new HashMap<>();
        for (char c : s.toCharArray()) {
            frequency.put(c, frequency.getOrDefault(c, 0) + 1);
        }
        return frequency;
    }
}
```
Output:
Character Frequencies: {a=1, g=2, i=1, m=2, n=1, o=1, p=1, r=2}

100. Convert a String to a Map of Characters and Their Positions
```
import java.util.HashMap;
import java.util.Map;
public class CharacterPositionMap {
    public static void main(String[] args) {
        String str = "hello";
        Map<Character, Integer> positionMap = characterPositions(str);
        System.out.println("Character Positions: " + positionMap);
    }
public static Map<Character, Integer> characterPositions(String s) {
        Map<Character, Integer> positionMap = new HashMap<>();
        for (int i = 0; i < s.length(); i++) {
            positionMap.put(s.charAt(i), i);
        }
        return positionMap;
    }
}
```
Output:
Character Positions: {h=0, e=1, l=2, o=4}

C programming -

String -
```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int length(char st[]) {
    int i = 0;
    while (st[i] != '\0') {
        i++;
    }
    return i;
}
void toLower(char st[6]){
        int i=0;
        while(st[i]!='\0'){
                if(st[i]>=65 && st[i]<=90)
                {
                        st[i] = st[i]+32;
                }
                i++;
        }
}
void copy(char st[], char str[]){
        int i=0;
        int len = strlen(str);
        while(i<=len){
                st[i] = str[i];
                i++;
```

```
        }
}
int main() {
        char st[6] = "";
        char str[6] = "Rishi";
        copy(st, str);
        printf("%s",st);
    return 0;
}
```

length -

```
#include <stdio.h>
#include <stdlib.h> // Include the standard library header for size_t
int length(char st[]) {
    int i = 0;
    while (st[i] != '\0') {
        i++;
    }
    return i;
}
int main() {
    char st[] = "Rishi singh";
    int len = length(st);
    printf("Length of the string: %d\n", len); // Use %d for integers
    return 0;
}
```

Level 1

Problems	Solve
Reverse words in a given string	Solve
Longest Common Prefix	Solve
Roman Number to Integer	Solve
Integer to Roman	Solve
Closest Strings	Solve
Divisible by 7	Solve
Encrypt the String – II	Solve
Equal point in a string of brackets	Solve
Isomorphic Strings	Solve
Check if two strings are k-anagrams or not	Solve
Panagram Checking	Solve
Minimum Deletions	Solve
Number of Distinct Subsequences	Solve
Check if string is rotated by two places	Solve

Level 2

Problems	Solve
Implement Atoi	Solve
Validate an IP address	Solve
License Key Formatting	Solve
Find the largest word in dictionary	Solve
Equal 0,1, and 2	Solve
Find and replace in String	
Add Binary Strings	Solve
Sum of two large numbers	Solve
Multiply two strings	Solve

Look and say Pattern	Solve
Minimum times A has to be repeated to make B a Substring	Solve
Excel Sheet – I	Solve
Form a Palindrome	Solve
Find the N-th character	Solve
Next higher palindromic number using the same set of digits	Solve
Length of longest prefix suffix	Solve
Longest K unique characters substring	Solve
Smallest window in string containing all characters	Solve
Longest Palindromic Subsequence	Solve
Longest substring without repeating characters	Solve
Substrings of length k with k-1 distinct elements	Solve
Count number of substrings	Solve
Interleaved Strings	Solve
Print Anagrams together	Solve
Rank the permutation	Solve
A Special Keyboard	Solve

Level 3

Problems	**Solve**
Restrictive Candy Crush	Solve
Edit Distance	Solve
Search Pattern (KMP-Algorithm)	Solve
Search Pattern (Rabin-Karp Algorithm)	Solve
Search Pattern (Z-algorithm)	Solve
Shortest Common Supersequence	Solve
Number of words with K maximum distinct vowels	Solve
Longest substring to form a Palindrome	Solve
Longest Valid Parenthesis	Solve
Distinct Palindromic Substrings	Solve

Related Articles:

- Top 50 Array Coding Problems for Interviews
- Top 50 Tree Coding Problems for Interviews
- Top 50 Graph Coding Problems for Interviews
- Top 50 Dynamic Programming Coding Problems for Interviews
- Top 50 Sorting Coding Problems for Interviews
- Top 50 Searching Coding Problems for Interviews
- Top 50 Binary Search Tree Coding Problems for Interviews