Master Coding Interview Questions (+DSA)

P.S -You can convert the code to Python also by simple prompt in ChatGPT

https://chat.openai.com/share/884ea526-e131-4b0b-a63e-d2ba87d95f11

Prompt - Convert the below program to Python 3.11 version "Paste Code"

✅ How to Take Input from Users?

- You can take input from users using the Scanner class

```
import java.util.Scanner;

public class UserInputExample {
    public static void main(String[] args) {
        // Create a Scanner object to read user input
        Scanner scanner = new Scanner(System.in);

        // Prompt the user for input
        System.out.print("Enter your name: ");
        // Read a line of text entered by the user
        String name = scanner.nextLine();
        // Prompt the user for a number
        System.out.print("Enter your age: ");
        // Read an integer entered by the user
        int age = scanner.nextInt();
        // Display the user's input
        System.out.println("Hello, " + name + "! You are " + age + "
years old.");
        // Close the Scanner to release resources (optional)
        scanner.close();
    }
}
```

✅ Table Print using System.out.printf

Print lines of output; each line (where) contains the of in the form:

`N x i = result.`

Sample Input

2

Sample Output

-
- 2 x 1 = 2
- 2 x 2 = 4
- 2 x 3 = 6
- 2 x 4 = 8
- 2 x 5 = 10
- 2 x 6 = 12
- 2 x 7 = 14
- 2 x 8 = 16
- 2 x 9 = 18
- 2 x 10 = 20

```
int n = 5;
for (int i = 1; i <= 10; i++) {
  System.out.printf("%d\t%d\t\n", i+”x”+n+”=”, n*i);
}
```

✅ FizzBuzz Test:

Write a program that prints numbers from 1 to 100. However, for multiples of 3, print "Fizz" instead of the number, and for multiples of 5, print "Buzz." For numbers that are multiples of both 3 and 5, print "FizzBuzz."

```
public class FizzBuzz {
    public static void main(String[] args) {
        for (int i = 1; i <= 100; i++) {
            if (i % 3 == 0 && i % 5 == 0) {
                System.out.println("FizzBuzz");
            } else if (i % 3 == 0) {
                System.out.println("Fizz");
            } else if (i % 5 == 0) {
                System.out.println("Buzz");
            } else {
                System.out.println(i);
            }
        }
    }
}
```

✅ Palindrome Checker:

Create a program that checks whether a given string is a palindrome. A palindrome is a word or phrase that reads the same backward as forward (ignoring spaces, punctuation, and capitalization). Use an if-else statement to determine if the string is a palindrome.

✅ Grade Calculator:

Write a program that calculates and displays the letter grade for a given numerical score (e.g., A, B, C, D, or F) based on the following grading scale:

A: 90-100
B: 80-89
C: 70-79
D: 60-69
F: 0-59

```
public class GradeCalculator {
    public static void main(String[] args) {
        int score = 85; // Replace with your numerical score
        char grade;

        if (score >= 90 && score <= 100) {
            grade = 'A';
        } else if (score >= 80 && score < 90) {
            grade = 'B';
        } else if (score >= 70 && score < 80) {
            grade = 'C';
        } else if (score >= 60 && score < 70) {
            grade = 'D';
        } else {
            grade = 'F';
        }

        System.out.println("Your grade is: " + grade);
    }
}
```

✅ Leap Year Checker:

Create a program that determines whether a given year is a leap year. A leap year is divisible by 4, but not by 100 unless it is also divisible by 400. Use an if-else statement to make this determination.

```
public class LeapYearChecker {
    public static void main(String[] args) {
        int year = 2024; // Replace with the year you want to check
```

```
        boolean isLeapYear = false;

        if ((year % 4 == 0 && year % 100 != 0) || (year % 400 == 0)) {
            isLeapYear = true;
        }

        if (isLeapYear) {
            System.out.println(year + " is a leap year.");
        } else {
            System.out.println(year + " is not a leap year.");
        }
    }
}
```

✅ Triangle Classifier:

Write a program that classifies a triangle based on its side lengths. Given three input values representing the lengths of the sides, determine if the triangle is equilateral (all sides are equal), isosceles (exactly two sides are equal), or scalene (no sides are equal). Use an if-else statement to classify the triangle.

```
public class TriangleClassifier {
    public static void main(String[] args) {
        int side1 = 5; // Replace with the lengths of your triangle's
sides
        int side2 = 4;
        int side3 = 4;

        if (side1 == side2 && side2 == side3) {
            System.out.println("Equilateral triangle");
        } else if (side1 == side2 || side1 == side3 || side2 == side3) {
            System.out.println("Isosceles triangle");
        } else {
            System.out.println("Scalene triangle");
        }
    }
}
```

✅ Right Triangle Star Pattern

```
*
**
***
****
*****
```

```
int n = 5;
for (int i = 1; i <= n; i++) {
    for (int j = 1; j <= i; j++) {
        System.out.print("*");
    }
    System.out.println();
}
```

✅ Left Triangle Star Pattern

```
*****
****
***
**
*
```

```
int n = 5;
for (int i = n; i >= 1; i--) {
    for (int j = 1; j <= i; j++) {
        System.out.print("*");
    }
    System.out.println();
}
```

✅ Pyramid pattern in Java

```
   *
  ***
 *****
*******
*********
```

```java
public class PyramidPattern {
    public static void main(String[] args) {
        int rows = 5; // Number of rows in the pyramid

        for (int i = 1; i <= rows; i++) {
            // Print spaces before the stars
            for (int j = 1; j <= rows - i; j++) {
                System.out.print(" ");
            }

            // Print stars
            for (int k = 1; k <= 2 * i - 1; k++) {
                System.out.print("*");
            }

            // Move to the next line
            System.out.println();
        }
    }
}
```

✅ Count vowels and consonants in a String

```java
public int[] countVowelsCons(String str) {
  int vCount = 0;
  int cCount = 0;

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

  return new int[]{vCount, cCount};
}
```

✅Prime Nuber checker

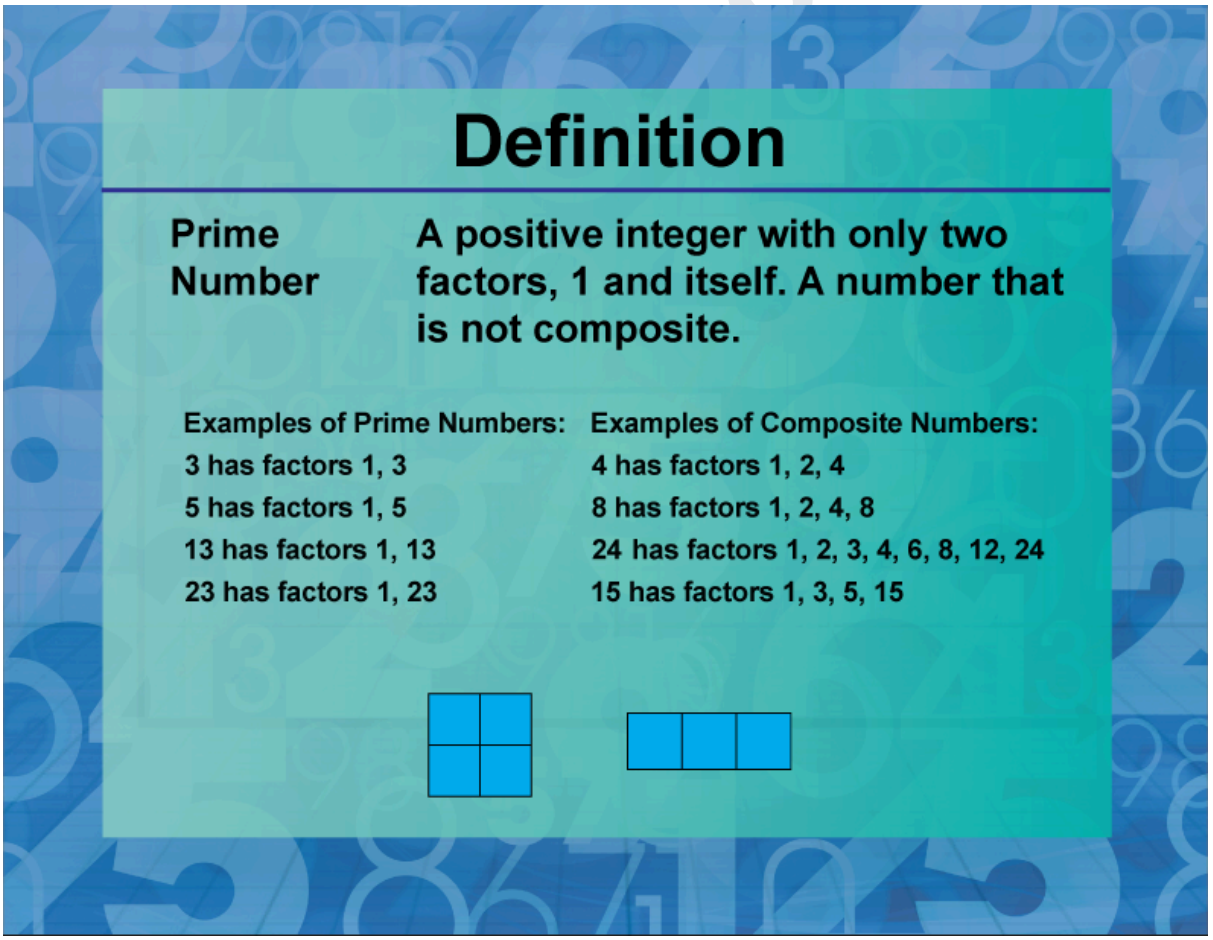
some valid and invalid test cases for the prime number checker in Java:

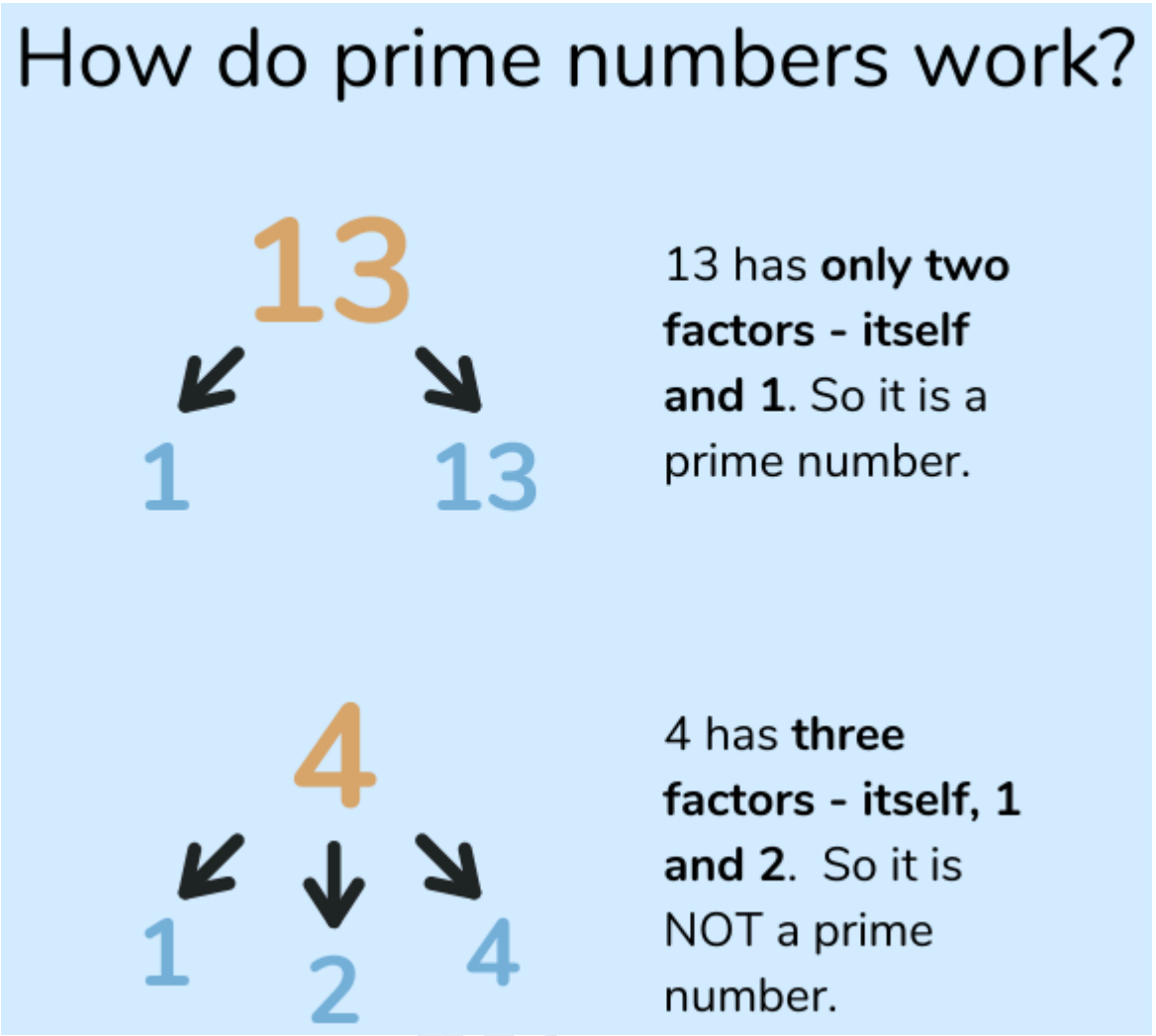
Valid Test Cases (Prime Numbers):

1. `num = 2` - This is the smallest prime number.
2. `num = 7` - A prime number greater than 2.
3. `num = 13` - Another prime number.
4. `num = 97` - A larger prime number.
5. `num = 101` - Another example of a prime number.

Invalid Test Cases (Non-Prime Numbers):

1. `num = 1` - 1 is not considered a prime number.
2. `num = 4` - It's divisible by 2, so it's not prime.
3. `num = 8` - Also divisible by 2.
4. `num = 15` - Divisible by 3 and 5.
5. `num = 100` - Divisible by 2 and 5.






```java
import java.util.Scanner;

public class PrimeChecker {
    public static boolean isPrime(int number) {
        if (number <= 1) {
            return false;
        }
        if (number <= 3) {
            return true;
        }
        if (number % 2 == 0 || number % 3 == 0) {
            return false;
        }
        for (int i = 5; i * i <= number; i += 6) {
            if (number % i == 0 || number % (i + 2) == 0) {
                return false;
            }
        }
```

```
        return true;
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a number: ");
        int num = scanner.nextInt();
        scanner.close();

        if (isPrime(num)) {
            System.out.println(num + " is a prime number.");
        } else {
            System.out.println(num + " is not a prime number.");
        }
    }
}
```

✅Palidrome of String

```
public class PalindromeCheck {

  public static boolean isPalindrome(String str) {

    str = str.toLowerCase();

    for(int i=0; i<str.length()/2; i++) {
      char start = str.charAt(i);
      char end = str.charAt(str.length()-1-i);

      if(start != end) {
        return false;
      }
    }
    return true;
  }

  public static void main(String[] args) {
    String input = "madam";
    if(isPalindrome(input)) {
      System.out.println("Palindrome");
```

```
    } else {
      System.out.println("Not Palindrome");
    }
  }

}
```

✅ String Reverse

Given a string , Reverse it and print true if they are same.
Madam is palindrome, pramod is not

Here is the full program to reverse words in-place in a given sentence using user input:

```java
import java.util.Scanner;

public class Main {

  public static void reverseWords(char[] sentence) {
    // Reverse entire sentence
    reverse(sentence, 0, sentence.length-1);

    int start = 0;
    for(int end=0; end<sentence.length; end++) {
      if(sentence[end] == ' ') {
        // Reverse current word
        reverse(sentence, start, end-1);
        start = end + 1;
      }
    }

    // Reverse last word
    reverse(sentence, start, sentence.length-1);
  }

  public static void reverse(char[] arr, int i, int j) {
    while(i < j) {
      char temp = arr[i];
      arr[i] = arr[j];
```

```
      arr[j] = temp;
      i++;
      j--;
    }
  }
```

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

    Scanner scanner = new Scanner(System.in);
    System.out.print("Enter a sentence: ");
    String input = scanner.nextLine();

    char[] sentence = input.toCharArray();
    reverseWords(sentence);

    System.out.println(new String(sentence));
  }
}
```

```

**Explanation:**

- Convert sentence to char array
- Reverse entire sentence first
- Then reverse each word by detecting word boundaries
- Reverse last word separately after loop
- Print reversed char array

This reverses the words in-place without using extra space.

Let me know if you have any other questions!

```
import java.util.Scanner;

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

    Scanner scanner = new Scanner(System.in);

    System.out.print("Enter a string: ");
    String input = scanner.nextLine();
```
```

```
    String reversed = new StringBuilder(input).reverse().toString();

    if(input.equals(reversed)) {
      System.out.println("Yes");
    } else {
      System.out.println("No");
    }

  }
}
```

✅ Duplicate String

```
public class RemoveDuplicates {

  public static String removeDuplicates(String str) {

    String result = "";

    for(int i=0; i<str.length(); i++) {
      char ch = str.charAt(i);
      if(result.indexOf(ch) == -1) {
        result += ch;
      }
    }

    return result;
  }

  public static void main(String[] args) {
    String input = "Hello World";
    String output = removeDuplicates(input);
    System.out.println(output);
  }

}
```

Advance using LinkedHashSet

```java
public class RemoveDuplicates {

  public static String remove(String str) {
    // add chars to LinkedHashSet
  }

  public static void main(String[] args) {
    String input = "Hello World";
    String output = remove(input);
    System.out.println(output); // Helo Wrd
  }

}
```

✅ Java Anagrams

```java
String s1 = "silent";
String s2 = "listen";

if (areAnagrams(s1, s2)) {
  System.out.println("Anagrams");
} else {
  System.out.println("Not Anagrams");
}
```

```java
public static boolean areAnagrams(String s1, String s2) {

  // Remove non-letter characters and convert to lowercase
  s1 = s1.replaceAll("[^a-zA-Z]", "").toLowerCase();
  s2 = s2.replaceAll("[^a-zA-Z]", "").toLowerCase();

  // Check if lengths are different
  if (s1.length() != s2.length()) {
    return false;
  }

  // Sort the characters of the strings
  char[] arr1 = s1.toCharArray();
  Arrays.sort(arr1);
  char[] arr2 = s2.toCharArray();
  Arrays.sort(arr2);
```

```
  // Compare sorted strings
  for (int i = 0; i < arr1.length; i++) {
    if (arr1[i] != arr2[i]) {
      return false;
    }
  }

  return true;
}
```

Another approach

```
public static boolean areAnagrams(String s1, String s2){

  // Remove non-letters and convert to same case
  s1 = s1.replaceAll("[^a-zA-Z]", "").toLowerCase();
  s2 = s2.replaceAll("[^a-zA-Z]", "").toLowerCase();

  // Check if lengths are different
  if(s1.length() != s2.length()){
    return false;
  }

  // Count frequency of characters
  int[] charFreq = new int[26];
  for(int i=0; i<s1.length(); i++){
    charFreq[s1.charAt(i) - 'a']++;
    charFreq[s2.charAt(i) - 'a']--;
  }

  // Check if all counts are 0
  for(int count : charFreq){
    if(count != 0){
      return false;
    }
  }
  return true;
}
```

✅ Valid Email Regex Java

pramod@live.com - Valid
pramod@dasda - Not valid

```
import java.util.regex.Pattern;
import java.util.regex.Matcher;

public class EmailValidation {

  public static void main(String[] args) {

    String email = "john@example.com";

    String emailRegex = "^[a-zA-Z0-9_+&*-]+(?:\\."+
                        "[a-zA-Z0-9_+&*-]+)*@" +
                        "(?:[a-zA-Z0-9-]+\\.)+[a-z" +
                        "A-Z]{2,7}$";

    Pattern pat = Pattern.compile(emailRegex);
    Matcher mat = pat.matcher(email);

    if (mat.matches())
      System.out.println("Valid email");
    else
      System.out.println("Invalid email");

  }
}
```

Test for Invalid Emails

- "plainaddress" - Missing @ and domain
- "@invalid.com" - Missing username
- "joe@[123.123.123.123]" - Square brackets around IP address
- " näme@example.com" - Contains illegal character
- "john..doe@example.com" - Double dot
- "john@doe@example.com" - Double @
- "john@example@com" - Missing top level domain (.com)
- "john@example.c" - Top level domain too short
- "john@example.com1" - Top level domain cannot have numbers
- "<john@example..com>" - Double dot in domain name
- "john@example.com." - Trailing dot in domain name
- "<john@example..com>." - Double dots and trailing dot
- "john@example.com.." - Double dot before top level domain

✅ Exceptions

✅ File Read and Token it

✅ List Questions

✅ Hash map

✅ Hashtable

✅ Design Patterns

✅ Count vowels and consonants in a String

aeiou
input = pramod
vol = 2

```
# Function to count vowels and consonants in a string
def count_vowels_and_consonants(input_string):
    # Convert the input string to lowercase to handle both uppercase
and lowercase characters
    input_string = input_string.lower()

    # Initialize counters for vowels and consonants
    vowel_count = 0
    consonant_count = 0

    # Define a set of vowels
    vowels = "aeiou"

    # Iterate through each character in the input string
    for char in input_string:
        if char.isalpha():
            if char in vowels:
                vowel_count += 1
            else:
                consonant_count += 1

    return vowel_count, consonant_count
```

```
# Input string
input_string = "pramod"

# Call the function to count vowels and consonants
vowel_count, consonant_count =
count_vowels_and_consonants(input_string)

# Print the results
print("Vowel Count:", vowel_count)
print("Consonant Count:", consonant_count)
```

✅ Leetcode 15 Easy Challenges

15 easy LeetCode problems along with their solutions in Java:

1. Two Sum

Problem Statement:
Given an array of integers, return indices of the two numbers such that they add up to a specific target.

Java Solution:

```
public int[] twoSum(int[] nums, int target) {
    Map<Integer, Integer> map = new HashMap<>();
    for (int i = 0; i < nums.length; i++) {
        int complement = target - nums[i];
        if (map.containsKey(complement)) {
            return new int[] { map.get(complement), i };
        }
        map.put(nums[i], i);
    }
    throw new IllegalArgumentException("No two sum solution");
}
```

2. Reverse Integer

Problem Statement:
Reverse digits of an integer.

Java Solution:

```
public int reverse(int x) {
    int result = 0;
    while (x != 0) {
        int pop = x % 10;
        x /= 10;
        if (result > Integer.MAX_VALUE / 10 || (result ==
Integer.MAX_VALUE / 10 && pop > 7)) {
            return 0;
        }
        if (result < Integer.MIN_VALUE / 10 || (result ==
Integer.MIN_VALUE / 10 && pop < -8)) {
            return 0;
        }
        result = result * 10 + pop;
    }
    return result;
}
```

3. Palindrome Number

Problem Statement:
Determine whether an integer is a palindrome.

Java Solution:

```
public boolean isPalindrome(int x) {
    if (x < 0) return false;
    int original = x;
    int reversed = 0;
    while (x != 0) {
        int digit = x % 10;
        reversed = reversed * 10 + digit;
        x /= 10;
    }
    return original == reversed;
}
```

4. Roman to Integer

Problem Statement:
Convert a Roman numeral to an integer.

Java Solution:

```
public int romanToInt(String s) {
    Map<Character, Integer> values = new HashMap<>();
    values.put('I', 1);
    values.put('V', 5);
    values.put('X', 10);
    values.put('L', 50);
    values.put('C', 100);
    values.put('D', 500);
    values.put('M', 1000);

    int result = 0;
    for (int i = 0; i < s.length(); i++) {
        int currentVal = values.get(s.charAt(i));
        int nextVal = (i < s.length() - 1) ? values.get(s.charAt(i + 1))
: 0;
        if (currentVal < nextVal) {
            result -= currentVal;
        } else {
            result += currentVal;
        }
    }
    return result;
}
```

5. Longest Common Prefix

Problem Statement:
Find the longest common prefix string amongst an array of strings.

Java Solution:

```
public 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;
}
```

6. Valid Parentheses

Problem Statement:
Given a string containing just the characters '(', ')', '{', '}', '[' and ']', determine if the input string is valid.

Java Solution:

```
public boolean isValid(String s) {
    Stack<Character> stack = new Stack<>();
    for (char c : s.toCharArray()) {
        if (c == '(' || c == '{' || c == '[') {
            stack.push(c);
        } else if (c == ')' && !stack.isEmpty() && stack.peek() == '(')
{
            stack.pop();
        } else if (c == '}' && !stack.isEmpty() && stack.peek() == '{')
{
            stack.pop();
        } else if (c == ']' && !stack.isEmpty() && stack.peek() == '[')
{
            stack.pop();
        } else {
            return false;
        }
    }
    return stack.isEmpty();
}
```

7. Merge Two Sorted Lists

Problem Statement:
Merge two sorted linked lists and return it as a new sorted list.

Java Solution:

```
public ListNode mergeTwoLists(ListNode l1, ListNode l2) {
    if (l1 == null) return l2;
```

```
    if (l2 == null) return l1;

    if (l1.val < l2.val) {
        l1.next = mergeTwoLists(l1.next, l2);
        return l1;
    } else {
        l2.next = mergeTwoLists(l1, l2.next);
        return l2;
    }
}
```

8. Remove Duplicates from Sorted Array

Problem Statement:
Given a sorted array, remove the duplicates in-place such that each element appears only once and return the new length.

Java Solution:

```
public int removeDuplicates(int[] nums) {
    if (nums.length == 0) return 0;
    int uniqueCount = 1;
    for (int i = 1; i < nums.length; i++) {
        if (nums[i] != nums[i - 1]) {
            nums[uniqueCount] = nums[i];
            uniqueCount++;
        }
    }
    return uniqueCount;
}
```

9. Implement strStr()

Problem Statement:
Implement strStr() which locates a substring within a given string.

Java Solution:

```
public int strStr(String haystack, String needle) {
    if (needle.isEmpty()) return 0;
    for (int i = 0; i <= haystack.length() - needle.length(); i++) {
        if (haystack.substring(i, i + needle.length()).equals(needle)) {
            return i;
```

```
        }
    }
    return -1;
}
```

10. Count and Say

Problem Statement:
The count-and-say sequence is the sequence of integers with the first five terms as follows: 1, 11, 21, 1211, 111221, ...

Java Solution:

```
public String countAndSay(int n) {
    if (n <= 0) return "";
    String result = "1";
    for (int i = 1; i < n; i++) {
        StringBuilder sb = new StringBuilder();
        char prevChar = result.charAt(0);
        int count = 1;
        for (int j = 1; j < result.length(); j++) {
            char currentChar = result.charAt(j);
            if (currentChar == prevChar) {
                count++;
            } else {
                sb.append(count).append(prevChar);
                prevChar = currentChar;
                count = 1;
            }
        }
        sb.append(count).append(prevChar);
        result = sb.toString();
    }
    return result;
}
```

11. Maximum Subarray

Problem Statement:
Find the contiguous subarray (containing at least one number) which has the largest sum and return its sum.

Java Solution:

```java
public int maxSubArray(int[] nums) {
    int maxSum = nums[0];
    int currentSum = nums[0];
    for (int i = 1; i < nums.length; i++) {
        currentSum = Math.max(nums[i], currentSum + nums[i]);
        maxSum = Math.max(maxSum, currentSum);
    }
    return maxSum;
}
```

12. Climbing Stairs

Problem Statement:
You are climbing a staircase. It takes n steps to reach the top. Each time you can either climb 1 or 2 steps. In how many distinct ways can you climb to the top?

Java Solution:

```java
public int climbStairs(int n) {
    if (n <= 2) return n;
    int prev1 = 1;
    int prev2 = 2;
    for (int i = 3; i <= n; i++) {
        int current = prev1 + prev2;
        prev1 = prev2;
        prev2 = current;
    }
    return prev2;
}
```

13. Plus One

Problem Statement:
Given a non-empty array of digits representing a non-negative integer, increment one to the integer.

Java Solution:

```java
public int[] plusOne(int[] digits) {
    int n = digits.length;
    for (int i = n - 1; i >= 0; i--) {
```

```
        if (digits[i] < 9) {
            digits[i]++;
            return digits;
        }
        digits[i] = 0;
    }
    int[] newNumber = new int[n + 1];
    newNumber[0] = 1;
    return newNumber;
}
```

14. Add Binary

Problem Statement:
Given two binary strings, return their sum (also a binary string).

Java Solution:

```
public String addBinary(String a, String b) {
    StringBuilder result = new StringBuilder();
    int carry = 0;
    int i = a.length() - 1, j = b.length() - 1;
    while (i >= 0 || j >= 0 || carry > 0) {
        int sum = carry;
        if (i >= 0) {
            sum += a.charAt(i--) - '0';
        }
        if (j >= 0) {
            sum += b.charAt(j--) - '0';
        }
        result.insert(0, sum % 2);
        carry = sum / 2;
    }
    return result.toString();
}
```

15. Sqrt(x)

Problem Statement:
Compute and return the square root of x, where x is a non-negative integer.

Java Solution:

```java
public int mySqrt(int x) {
    if (x == 0) return 0;
    int left = 1, right = x;
    while (left <= right) {
        int mid = left + (right - left) / 2;
        if (mid <= x / mid && (mid + 1) > x / (mid + 1)) {
            return mid;
        } else if (mid > x / mid) {
            right = mid - 1;
        } else {
            left = mid + 1;
        }
    }
    return -1; // This should not happen for non-negative x.
}
```