Q1. Reverse Integer

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

    public static void main(String[] args) {
        int x = 123;
        System.out.println("Reversed: " + reverse(x)); // Output: 321

        x = -123;
        System.out.println("Reversed: " + reverse(x)); // Output: -321

        x = 120;
        System.out.println("Reversed: " + reverse(x)); // Output: 21

        x = 0;
        System.out.println("Reversed: " + reverse(x)); // Output: 0
    }
}
```

Q2. Removing vowels from a given string

```
import java.util.Arrays;
import java.util.List;

class Main {
    static String remVowel(String str) {
        return str.replaceAll("[aeiouAEIOU]", "");
    }
    // Driver Code
    public static void main(String[] args) {
        String str = "Prepinsta";
        System.out.println(remVowel(str));
    }

}
```

Q3. Sum of Prime Factors

```
import java.util.Scanner;

public class PrimeFactorSum {
    public static boolean isPrime(int num) {
        if (num <= 1) {
            return false;
        }
        for (int i = 2; i * i <= num; i++) {
            if (num % i == 0) {
                return false;
            }
        }
        return true;
    }

    public static int sumOfPrimeFactors(int num) {
        int sum = 0;
        for (int i = 2; i <= num; i++) {
            while (num % i == 0 && isPrime(i)) {
                sum += i;
                num /= i;
            }
        }
        if (num > 1 && isPrime(num)) {
            sum += num;
        }
        return sum;
    }

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

        int sum = sumOfPrimeFactors(number);
        System.out.println("Sum of prime factors of " + number + " is: "
+ sum);
    }
}
```

Q4. Encryption By Digits (lowercase)

```
public class EncryptByDigits {
    public static String encryptByDigits(String text) {
        StringBuilder encryptedText = new StringBuilder();

        for (char ch : text.toCharArray()) {
            if (Character.isLetter(ch)) {
```

```
                encryptedText.append((int) ch - 96); // Convert
alphabetic character to its corresponding number
            } else {
                encryptedText.append(ch); // Keep non-alphabetic
characters unchanged
            }
        }

        return encryptedText.toString();
    }

    public static void main(String[] args) {
        String text = "hello world";
        String encryptedText = encryptByDigits(text);
        System.out.println(encryptedText);  // Output: "851212 2315187"
    }
}
```

Q5. Count the no. Of Occurrences of digit 9 in the array of given numbers

```
public class CountDigitOccurrences {
    public static int countDigitOccurrences(int[] numbers, int digit) {
        int count = 0;
        String digitStr = Integer.toString(digit);

        for (int number : numbers) {
            String numberStr = Integer.toString(number);
            for (char ch : numberStr.toCharArray()) {
                if (Character.toString(ch).equals(digitStr)) {
                    count++;
                }
            }
        }

        return count;
    }

    public static void main(String[] args) {
        int[] numbers = {909, 129, 493, 981, 1009};
        int digit = 9;
        int occurrences = countDigitOccurrences(numbers, digit);
        System.out.println("Number of occurrences of digit " + digit + ":
" + occurrences);
    }
}
```

Q6. Difference between highest and lowest prime numbers in an list

```
import java.util.Arrays;
import java.util.List;

public class PrimeDifference {
    public static int
differenceBetweenHighestAndLowestPrime(List<Integer> numbers) {
        Integer minPrime = null;
        Integer maxPrime = null;

        for (int num : numbers) {
            if (isPrime(num)) {
                if (minPrime == null || num < minPrime) {
                    minPrime = num;
                }
                if (maxPrime == null || num > maxPrime) {
                    maxPrime = num;
                }
            }
        }

        if (minPrime == null || maxPrime == null) {
            return 0;
        }

        return maxPrime - minPrime;
    }

    private static boolean isPrime(int num) {
        if (num <= 1) return false;
        if (num <= 3) return true;
        if (num % 2 == 0 || num % 3 == 0) return false;
        for (int i = 5; i * i <= num; i += 6) {
            if (num % i == 0 || num % (i + 2) == 0) return false;
        }
        return true;
    }

    public static void main(String[] args) {
        List<Integer> numbers = Arrays.asList(10, 2, 3, 5, 8, 13, 17);
        int difference = differenceBetweenHighestAndLowestPrime(numbers);
        System.out.println("Difference between highest and lowest prime:
" + difference); // Output: 15

        numbers = Arrays.asList(4, 6, 8, 9);
        difference = differenceBetweenHighestAndLowestPrime(numbers);
        System.out.println("Difference between highest and lowest prime:
" + difference); // Output: 0
    }
}
```

Q7. String palindrome

```
import java.util.Scanner;

public class StringIsAPalindromeOrNot {

      public static void main(String[] args) {

            String s = "arora";
            String rev = "";
            for (int i = s.length()-1; i >=0 ; i--)
                  rev=rev+s.charAt(i);
            if(s.equals(rev))
                  System.out.println("String is palindrome");
            else
                  System.out.println("String is not palindrome");

      }

}
```

Q8. Prime number

```
public class PrimeExample{
public static void main(String args[]){
  int i,m=0,flag=0;
  int n=3;//it is the number to be checked
  m=n/2;
  if(n==0||n==1){
   System.out.println(n+" is not prime number");
  }else{
   for(i=2;i<=m;i++){
    if(n%i==0){
     System.out.println(n+" is not prime number");
     flag=1;
     break;
    }
   }
   if(flag==0)  { System.out.println(n+" is prime number"); }
  }//end of else
}
}
```

Q9. count the number of words in a list that start with the letter 'A' or 'a'

```
import java.util.Arrays;
import java.util.List;

public class CountWordsStartingWithA {
    public static int countWordsStartingWithA(List<String> words) {
        int count = 0;
```

```
        for (String word : words) {
            if (word != null && !word.isEmpty() && (word.charAt(0) == 'A'
|| word.charAt(0) == 'a')) {
                count++;
            }
        }

        return count;
    }

    public static void main(String[] args) {
        List<String> words = Arrays.asList("Apple", "banana", "Avocado",
"apricot", "grape", "Aardvark");

        int count = countWordsStartingWithA(words);
        System.out.println("Number of words starting with 'A' or 'a': " +
count);  // Output: 4
    }
}
```

Q10. count the occurrence of each character in a string

```
import java.util.Scanner;
public class CountOccuranceOfChar1
{
public static void main(String args[])
{
String str;
int i, len;
int counter[] = new int[256];
Scanner scanner = new Scanner(System.in);
System.out.print("Please enter a string: ");
//reading a string from the user
str = scanner.nextLine();
//finds the length of the string
len = str.length();
// loop through the string and count frequency of every character and
store it in counter array
for (i = 0; i < len; i++)
{
counter[(int) str.charAt(i)]++;
}
//print Frequency of characters
for (i = 0; i < 256; i++)
{
if (counter[i] != 0)
{
//prints frequency of characters
System.out.println((char) i + " --> " + counter[i]);
}
}
```

```
}
}
```

Q11. Circular array

```
import java.util.Scanner;
public class CountOccuranceOfChar1
{
public static void main(String args[])
{
String str;
int i, len;
int counter[] = new int[256];
Scanner scanner = new Scanner(System.in);
System.out.print("Please enter a string: ");
//reading a string from the user
str = scanner.nextLine();
//finds the length of the string
len = str.length();
// loop through the string and count frequency of every character and
store it in counter array
for (i = 0; i < len; i++)
{
counter[(int) str.charAt(i)]++;
}
//print Frequency of characters
for (i = 0; i < 256; i++)
{
if (counter[i] != 0)
{
//prints frequency of characters
System.out.println((char) i + " --> " + counter[i]);
}
}
}
}
```

Q12. Number of Steps to reduce a number to Zero

```
public int numberOfSteps ( int num) {
        int count = 0 ;
        while (num != 0 ) {
            count++;
            if (num % 2 == 0 ) {
                num = num / 2 ;
            } else {
                num--;
            }
        }
        return count;
```

```
    }
```

Q13. Reverse of a String

```
public class StringReverser {
    public static String reverseString(String text) {
        // Convert the string to a character array
        char[] charArray = text.toCharArray();
        int left = 0;
        int right = charArray.length - 1;

        // Swap characters from start and end until the middle is reached
        while (left < right) {
            // Swap characters
            char temp = charArray[left];
            charArray[left] = charArray[right];
            charArray[right] = temp;

            // Move towards the middle
            left++;
            right--;
        }

        // Convert the character array back to a string
        return new String(charArray);
    }

    public static void main(String[] args) {
        String text = "Hello, World!";
        String reversedText = reverseString(text);
        System.out.println("Original string: " + text);
        System.out.println("Reversed string: " + reversedText);
    }
}
```

Q14. Sum of Natural numbers

```
public class SumOfNaturalNumbers {

    // Function to calculate the sum of natural numbers up to n
    public static int sumOfNaturalNumbers(int n) {
        // Formula for sum of first n natural numbers: n * (n + 1) / 2
        return n * (n + 1) / 2;
    }

    public static void main(String[] args) {
        int n = 10; // Example value, you can change this to test other
numbers
        int sum = sumOfNaturalNumbers(n);
```

```
        System.out.println("Sum of natural numbers up to " + n + ": " +
sum);
    }
}
```

Q15. Count the number of perfect squares

```
public class PerfectSquareCounter {

    // Function to check if a number is a perfect square
    public static boolean isPerfectSquare(int num) {
        if (num < 0) {
            return false; // Negative numbers cannot be perfect squares
        }
        int sqrt = (int) Math.sqrt(num);
        return (sqrt * sqrt == num);
    }

    // Function to count perfect squares in an array
    public static int countPerfectSquares(int[] numbers) {
        int count = 0;
        for (int num : numbers) {
            if (isPerfectSquare(num)) {
                count++;
            }
        }
        return count;
    }

    public static void main(String[] args) {
        int[] numbers = {1, 4, 9, 16, 25, 30, 36, 50};
        int count = countPerfectSquares(numbers);
        System.out.println("Number of perfect squares: " + count);
    }
}
```

Q16. Minimum and maximum number in an array

```
public class MinMaxFinder {

    // Function to find the minimum and maximum numbers in an array
    public static int[] findMinMax(int[] numbers) {
        if (numbers == null || numbers.length == 0) {
            throw new IllegalArgumentException("Array cannot be null or
empty");
        }

        int min = numbers[0];
        int max = numbers[0];
```

```
        for (int num : numbers) {
            if (num < min) {
                min = num;
            }
            if (num > max) {
                max = num;
            }
        }

        return new int[]{min, max};
    }

    public static void main(String[] args) {
        int[] numbers = {3, 5, 7, 2, 8, -1, 4, 10};
        int[] result = findMinMax(numbers);
        System.out.println("Minimum number: " + result[0]);
        System.out.println("Maximum number: " + result[1]);
    }
}
```

Q17. Maximum LCM among all pairs

```
import java.util.*;
class GFG {

    // Function comparing all LCM pairs
    static int maxLcmOfPairs(int arr[], int n)
    {
        // To store the highest LCM
        int maxLCM = -1;

        // To generate all pairs from array
        for (int i = 0; i < n; i++) {
            for (int j = i + 1; j < n; j++) {

                // Find LCM of the pair
                // Update the maxLCM if this is
                // greater than its existing value
                maxLCM = Math.max(
                    maxLCM, (arr[i] * arr[j])
                                / __gcd(arr[i], arr[j]));
            }
        }

        // Return the highest value of LCM
        return maxLCM;
    }

    static int __gcd(int a, int b)
    {
        return b == 0 ? a : __gcd(b, a % b);
    }
```

```
    // Driver code
    public static void main(String[] args)
    {
        int arr[] = { 17, 3, 8, 6 };
        int n = arr.length;

        System.out.print(maxLcmOfPairs(arr, n));
    }
}
```

Q18. Count vowels in a String

```
// Java Program to Count Number of Vowels
// in a String in a iterative way

import java.io.*;

public class vowel {
     public static void main(String[] args)
           throws IOException
     {
           String str = "GeeksForGeeks";
           str = str.toLowerCase();
           int count = 0;

           for (int i = 0; i < str.length(); i++) {
                 // check if char[i] is vowel
                 if (str.charAt(i) == 'a' || str.charAt(i) == 'e'
                       || str.charAt(i) == 'i'
                       || str.charAt(i) == 'o'
                       || str.charAt(i) == 'u') {
                       // count increments if there is vowel in
                       // char[i]
                       count++;
                 }
           }

           // display total count of vowels in string
           System.out.println(
                 "Total no of vowels in string are: " + count);
     }
}
```

Q19. Finding Hidden Integer

```
// Java Program to find the
// hidden number

public class GFG {
```

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

            // Getting the size of array
            int n = 3;

            // Getting the array of size n
            int a[] = { 1, 2, 3 };

            // Solution
            int i = 0;

            // Finding sum of the array elements
            long sum = 0;
            for (i = 0; i < n; i++) {
                  sum += a[i];
            }

            // Dividing sum by size n
            long x = sum / n;

            // Print x, if found
            if (x * n == sum)
                  System.out.println(x);
            else
                  System.out.println("-1");
      }
}
```

Q20. Substring

```
// working of substring(int begIndex, int endIndex)

// Driver Class
public class Substr2 {
    // main function
    public static void main(String args[])
    {
        // Initializing String
        String Str = new String("Welcome to geeksforgeeks");

        // using substring() to extract substring
        // returns geeks
        System.out.print("The extracted substring  is : ");
        System.out.println(Str.substring(10, 16));
    }
}
```

Q21. Find the Count of Palindromic Substring

```
// Java program to find the count of palindromic sub-string
// of a string in it's ascending form

class GFG {

    final static int MAX_CHAR = 26;

// function to return count of palindromic sub-string
    static int countPalindrome(String str) {
        int n = str.length();
        int sum = 0;

        // calculate frequency
        int hashTable[] = new int[MAX_CHAR];
        for (int i = 0; i < n; i++) {
            hashTable[str.charAt(i) - 'a']++;
        }

        // calculate count of palindromic sub-string
        for (int i = 0; i < 26; i++) {
            if (hashTable[i] != 0) {
                sum += (hashTable[i] * (hashTable[i] + 1) / 2);
            }
        }

        // return result
        return sum;
    }

// driver program
    public static void main(String[] args) {
        String str = "anananananddd";

        System.out.println(countPalindrome(str));

    }
}
```

Q22. Reverse the string and remove the leading zeros

```
public class ReverseAndRemoveLeadingZeros {

    // Function to reverse a string
    public static String reverseString(String text) {
        if (text == null || text.isEmpty()) {
            return text;
        }

        // Convert the string to a character array
        char[] charArray = text.toCharArray();
```

```
        int left = 0;
        int right = charArray.length - 1;

        // Swap characters from start and end until the middle is reached
        while (left < right) {
            char temp = charArray[left];
            charArray[left] = charArray[right];
            charArray[right] = temp;
            left++;
            right--;
        }

        return new String(charArray);
    }

    // Function to remove leading zeros from a string
    public static String removeLeadingZeros(String text) {
        if (text == null || text.isEmpty()) {
            return text;
        }

        // Remove leading zeros using regular expression
        return text.replaceFirst("^0+(?!$)", "");
    }

    public static void main(String[] args) {
        String originalText = "0012304500";

        // Reverse the string
        String reversedText = reverseString(originalText);
        System.out.println("Reversed string: " + reversedText);

        // Remove leading zeros from the reversed string
        String result = removeLeadingZeros(reversedText);
        System.out.println("Result after removing leading zeros: " +
result);
    }
}
```

Q23. Copy of array in a New one

```
// To use Arrays.toString() method
import java.util.Arrays;

class Main {
    public static void main(String[] args) {
        int[] n1 = {2, 3, 12, 4, 12, -2};

        int[] n3 = new int[5];

        // Creating n2 array of having length of n1 array
        int[] n2 = new int[n1.length];
```

```
        // copying entire n1 array to n2
        System.arraycopy(n1, 0, n2, 0, n1.length);
        System.out.println("n2 = " + Arrays.toString(n2));

        // copying elements from index 2 on n1 array
        // copying element to index 1 of n3 array
        // 2 elements will be copied
        System.arraycopy(n1, 2, n3, 1, 2);
        System.out.println("n3 = " + Arrays.toString(n3));
    }
}
```

Q24. Sum of all adjacent elements

```
public class AdjacentSum {

    // Function to find the sum of adjacent elements in an array
    public static int[] sumOfAdjacentElements(int[] numbers) {
        if (numbers == null || numbers.length < 2) {
            throw new IllegalArgumentException("Array must contain at
least two elements.");
        }

        // Array to store the sum of adjacent elements
        int[] result = new int[numbers.length - 1];

        // Compute the sum of each pair of adjacent elements
        for (int i = 0; i < numbers.length - 1; i++) {
            result[i] = numbers[i] + numbers[i + 1];
        }

        return result;
    }

    public static void main(String[] args) {
        int[] numbers = {2, 4, 6, 8, 10};
        int[] sums = sumOfAdjacentElements(numbers);

        System.out.println("Sum of adjacent elements:");
        for (int sum : sums) {
            System.out.println(sum);
        }
    }
}
```

Q25. Count of pairs with sum k

```
class GfG {
```

```
    // Function to count all pairs whose sum
    // is equal to the given target value
    static int twoSum(int[] arr, int target) {
        int n = arr.length;
        int count = 0;

        // Iterate through each element in the array
        for (int i = 0; i < n; i++) {

            // For each element arr[i], check every
            // other element arr[j] that comes after it
            for (int j = i + 1; j < n; j++) {

                // Check if the sum of the current pair
                // equals the target
                if (arr[i] + arr[j] == target) {
                    count++;
                }
            }
        }
        return count;
    }

    public static void main(String[] args) {

        int[] arr = {1, 5, 7, -1, 5};
        int target = 6;

        // Call the twoSum function and print the result
        System.out.println(twoSum(arr, target));
    }
}
```

Q26. find the product of 2 numbers when their sum and difference are given

```
// Formula (S+D)Ã—(Sâˆ’D)/4
public class ProductOfNumbers {

    // Function to calculate the product of two numbers given their sum
and difference
    public static int findProduct(int sum, int difference) {
        // Calculate the product using the formula: (sum^2 -
difference^2) / 4
        return (int) ((Math.pow(sum, 2) - Math.pow(difference, 2)) / 4);
    }

    public static void main(String[] args) {
        int sum = 10;         // Example sum
        int difference = 2;   // Example difference

        int product = findProduct(sum, difference);
```

```
        System.out.println("Product of the two numbers: " + product);
    }
}
```

Q27. Captilalize first letter

```
public class ProductOfNumbers {

    // Function to calculate the product of two numbers given their sum
and difference
    public static int findProduct(int sum, int difference) {
        // Calculate the product using the formula: (sum^2 -
difference^2) / 4
        return (int) ((Math.pow(sum, 2) - Math.pow(difference, 2)) / 4);
    }

    public static void main(String[] args) {
        int sum = 10;         // Example sum
        int difference = 2;   // Example difference

        int product = findProduct(sum, difference);
        System.out.println("Product of the two numbers: " + product);
    }
}
```

Q28. Sum of palindromic Lenghths

```
public class PalindromicLengthsSum {

    // Function to check if a substring is a palindrome
    private static boolean isPalindrome(String s) {
        int left = 0;
        int right = s.length() - 1;
        while (left < right) {
            if (s.charAt(left) != s.charAt(right)) {
                return false;
            }
            left++;
            right--;
        }
        return true;
    }

    // Function to calculate the sum of the lengths of all palindromic
substrings
    public static int sumOfPalindromicLengths(String text) {
        int sum = 0;
        int length = text.length();

        // Iterate through all possible substrings
```

```
        for (int start = 0; start < length; start++) {
            for (int end = start + 1; end <= length; end++) {
                String substring = text.substring(start, end);
                if (isPalindrome(substring)) {
                    sum += substring.length();
                }
            }
        }

        return sum;
    }

    public static void main(String[] args) {
        String text = "ababa";
        int sum = sumOfPalindromicLengths(text);
        System.out.println("Sum of lengths of palindromic substrings: " +
sum);
    }
}
```

Q29. Count the Adjacent elements which are Divisible by n

```
public class AdjacentDivisible {

    // Function to find and print pairs of adjacent elements divisible by
n
    public static void findAdjacentDivisible(int[] numbers, int n) {
        if (numbers == null || numbers.length < 2) {
            System.out.println("Array must contain at least two
elements.");
            return;
        }

        boolean found = false;

        // Iterate through the array and check each adjacent pair
        for (int i = 0; i < numbers.length - 1; i++) {
            if (numbers[i] % n == 0 && numbers[i + 1] % n == 0) {
                System.out.println("Adjacent elements divisible by " + n
+ ": " +
                                   numbers[i] + " and " + numbers[i +
1]);
                found = true;
            }
        }

        if (!found) {
            System.out.println("No adjacent elements are divisible by " +
n);
        }
    }
```

```
    public static void main(String[] args) {
        int[] numbers = {6, 12, 15, 18, 24};
        int n = 6;
        findAdjacentDivisible(numbers, n);
    }
}
```

Q30. Removing all characters except numbers in a String and printing the last 10 digits in that string.

```
public class ExtractDigits {

    // Function to remove all non-numeric characters and return the last 10 digits
    public static String getLast10Digits(String text) {
        if (text == null) {
            return "";
        }

        // Remove all non-numeric characters
        StringBuilder numericString = new StringBuilder();
        for (char ch : text.toCharArray()) {
            if (Character.isDigit(ch)) {
                numericString.append(ch);
            }
        }

        // Convert StringBuilder to String
        String result = numericString.toString();

        // If there are fewer than 10 digits, return the entire string
        if (result.length() <= 10) {
            return result;
        }

        // Get the last 10 digits
        return result.substring(result.length() - 10);
    }

    public static void main(String[] args) {
        String input = "abc123def4567890ghi12345"; // Example input
        String last10Digits = getLast10Digits(input);
        System.out.println("Last 10 digits: " + last10Digits);
    }
}
```

Q31. Find the maximum and minimum differences between adjacent elements in an array

```
public class AdjacentDifferences {

    // Function to find the maximum and minimum difference between
adjacent elements
    public static int[] findMaxMinDifference(int[] numbers) {
        if (numbers == null || numbers.length < 2) {
            throw new IllegalArgumentException("Array must contain at
least two elements.");
        }

        int maxDiff = Integer.MIN_VALUE;
        int minDiff = Integer.MAX_VALUE;

        // Iterate through the array to find differences between adjacent
elements
        for (int i = 0; i < numbers.length - 1; i++) {
            int diff = Math.abs(numbers[i] - numbers[i + 1]);
            if (diff > maxDiff) {
                maxDiff = diff;
            }
            if (diff < minDiff) {
                minDiff = diff;
            }
        }

        return new int[]{maxDiff, minDiff};
    }

    public static void main(String[] args) {
        int[] numbers = {3, 10, 6, 8, 15}; // Example input
        int[] result = findMaxMinDifference(numbers);

        System.out.println("Maximum Difference: " + result[0]);
        System.out.println("Minimum Difference: " + result[1]);
    }
}
```

Q32. Shift one element towards right

```
public class ShiftRight {

    // Function to shift elements of the array one position to the right
    public static void shiftRight(int[] array) {
        if (array == null || array.length == 0) {
            System.out.println("Array is empty or null.");
            return;
        }

        // Store the last element
        int lastElement = array[array.length - 1];
```

```
        // Shift elements to the right
        for (int i = array.length - 1; i > 0; i--) {
            array[i] = array[i - 1];
        }

        // Place the last element in the first position
        array[0] = lastElement;
    }

    public static void main(String[] args) {
        int[] array = {1, 2, 3, 4, 5}; // Example input
        System.out.println("Original array: ");
        printArray(array);

        shiftRight(array);

        System.out.println("Array after shifting right: ");
        printArray(array);
    }

    // Helper function to print the array
    public static void printArray(int[] array) {
        for (int num : array) {
            System.out.print(num + " ");
        }
        System.out.println();
    }
}
```