Q1 : Given an integer, find out the sum of its digits using recursion.
Input: n= 1234
Output: 10
Explanation: 1+2+3+4=10

Ans:

```
public class SumOfDigits {
    public static int sumOfDigits(int num) {
        // Base case: If the number is less than 10, return the number itself.
        if (num < 10) {
            return num;
        } else {
            // Recursive case: Get the last digit and add it to the sum of the digits of the remaining
number.
            int lastDigit = num % 10;
            int remainingDigits = num / 10;
            return lastDigit + sumOfDigits(remainingDigits);
        }
    }

    public static void main(String[] args) {
        int num = 12345;
        int result = sumOfDigits(num);
        System.out.println("Sum of digits of " + num + " is: " + result);
    }
}
```

Q2: Given a number n. Find the sum of natural numbers till n but with alternate signs. That means if n = 5 then you have to return 1-2+3-4+5 = 3 as your answer.

Ans:

```
public class checkSquare {

    public static int sumAlter(int num, int sum) {
        if (num <= 0) {
            return sum;
        }

        if (num % 2 == 1) {
            sum = sum + num;
```

```
        } else {
            sum = sum - num;
        }

        return sumAlter(num - 1, sum);

    }

    public static void main(String[] args) {

        int num = 10;
        System.out.println(sumAlter(num, 0));
    }

}
```

Q3: Print the max value of the array [13, 1, -3, 22, 5].

Ans:

```
public class MaxValueRecursive {

    public static void main(String[] args) {
        int[] array = {13, 1, -3, 22, 5};
        int maxValue = findMaxValue(array, 0);
        System.out.println("The maximum value in the array is: " + maxValue);
    }

    public static int findMaxValue(int[] arr, int index) {
        if (index == arr.length - 1) {
            return arr[index];
        } else {
            int maxRest = findMaxValue(arr, index + 1);
            return Math.max(arr[index], maxRest);
        }
    }
}
```

Q4 : Find the sum of the values of the array [92, 23, 15, -20, 10].

Ans:

```
public class SumOfArray {

    public static void main(String[] args) {
        int[] array = {92, 23, 15, -20, 10};
        int sum = findSum(array, 0);
        System.out.println("The sum of the array elements is: " + sum);
    }

    public static int findSum(int[] arr, int index) {
        if (index == arr.length) {
            return 0; // Base case: When the index reaches the end of the array
        } else {
            int currentElement = arr[index];
            int sumOfRest = findSum(arr, index + 1);
            return currentElement + sumOfRest;
        }
    }
}
```

Q5. Given a number n. Print if it is an armstrong number or not. An armstrong number is a number if the sum of every digit in that number raised to the power of total digits in that number is equal to the number.
Example : 153 = 1^3 + 5^3 + 3^3 = 1 + 125 + 27 = 153 hence 153 is an armstrong number. (Easy)
Input1 : 153
Output1 : Yes
Input 2 : 134
Output2 : No

Ans:

```
public class ArmstrongNumberRecursive {

    public static void main(String[] args) {
        int number = 153; // Change this to the number you want to check

        if (isArmstrongNumber(number, countDigits(number))) {
            System.out.println(number + " is an Armstrong number.");
        } else {
```

```
            System.out.println(number + " is not an Armstrong number.");
        }
    }

    public static int countDigits(int num) {
        if (num == 0) {
            return 0;
        }
        return 1 + countDigits(num / 10);
    }

    public static boolean isArmstrongNumber(int num, int numberOfDigits) {
        int originalNumber = num;

        // Calculate the sum of each digit raised to the power of the total number of digits
        int sum = calculateArmstrongSum(num, numberOfDigits);

        // Check if the sum is equal to the original number
        return sum == originalNumber;
    }

    public static int calculateArmstrongSum(int num, int numberOfDigits) {
        if (num == 0) {
            return 0;
        }

        int digit = num % 10;
        return (int) (Math.pow(digit, numberOfDigits) + calculateArmstrongSum(num / 10,
numberOfDigits));
    }
}
```