Q1: Sum of Digits Using Recursion

```
public class SumOfDigits {
    public static int sumOfDigits(int n) {
        // Base case: If n is 0, return 0
        if (n == 0) {
            return 0;
        }
        // Recursive case: Add the last digit to the sum of the remaining digits
        return (n % 10) + sumOfDigits(n / 10);
    }

    public static void main(String[] args) {
        int n = 1234;
        System.out.println(sumOfDigits(n));  // Output: 10
    }
}
```

Q2: Sum of Natural Numbers Till `n` with Alternate Signs Using Recursion

```
public class SumWithAlternateSigns {
    public static int sumWithAlternateSigns(int n) {
        // Base case: If n is 0, return 0
        if (n == 0) {
            return 0;
        }
        // Recursive case: If n is even, subtract n; if odd, add n
        return (n % 2 == 0 ? -n : n) + sumWithAlternateSigns(n - 1);
    }

    public static void main(String[] args) {
        int n1 = 10;
        int n2 = 5;
        System.out.println(sumWithAlternateSigns(n1));  // Output: -5
        System.out.println(sumWithAlternateSigns(n2));  // Output: 3
    }
}
```

Q3: Print the Max Value of the Array Using Recursion

```
public class MaxValueInArray {
```

```
    public static int findMax(int[] arr, int n) {
        // Base case: If the array has only one element, return it
        if (n == 1) {
            return arr[0];
        }
        // Recursive case: Return the maximum of the last element and the maximum of the rest
        return Math.max(arr[n - 1], findMax(arr, n - 1));
    }

    public static void main(String[] args) {
        int[] arr = {13, 1, -3, 22, 5};
        System.out.println("Max value in the array is: " + findMax(arr, arr.length));  // Output: 22
    }
}
```

Q4: Find the Sum of the Values of the Array Using Recursion

```
public class SumOfArray {
    public static int sumOfArray(int[] arr, int n) {
        // Base case: If the array has no elements, return 0
        if (n == 0) {
            return 0;
        }
        // Recursive case: Add the last element to the sum of the rest
        return arr[n - 1] + sumOfArray(arr, n - 1);
    }

    public static void main(String[] args) {
        int[] arr = {92, 23, 15, -20, 10};
        System.out.println("Sum of the array values is: " + sumOfArray(arr, arr.length));  // Output:
120
    }
}
```

Q5: Check if a Number is an Armstrong Number Using Recursion

```
public class ArmstrongNumber {
    public static int power(int base, int exp) {
        // Base case: If exponent is 0, return 1
        if (exp == 0) {
            return 1;
```

```
        }
        // Recursive case: Multiply the base with the result of power(base, exp-1)
        return base * power(base, exp - 1);
    }

    public static int sumOfPowers(int n, int digits) {
        // Base case: If n is 0, return 0
        if (n == 0) {
            return 0;
        }
        // Recursive case: Add the power of the last digit to the sum of powers of the rest
        int digit = n % 10;
        return power(digit, digits) + sumOfPowers(n / 10, digits);
    }

    public static boolean isArmstrong(int n) {
        int digits = String.valueOf(n).length();
        return sumOfPowers(n, digits) == n;
    }

    public static void main(String[] args) {
        int n1 = 153;
        int n2 = 134;
        System.out.println(isArmstrong(n1) ? "Yes" : "No");  // Output: Yes
        System.out.println(isArmstrong(n2) ? "Yes" : "No");  // Output: No
    }
}
```