Assignment 1: Recursion - Java Solutions

1. Count the Number of Digits in a Number

```
class CountDigits {
    static int countDigits(int n) {
        n = Math.abs(n);
        if (n < 10) return 1;
        return 1 + countDigits(n / 10);
    }

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

2. Find the Maximum Element in an Array using Recursion

```
class MaxInArray {
    static int findMax(int[] arr, int n) {
        if (n == 1) return arr[0];
        return Math.max(arr[n - 1], findMax(arr, n - 1));
    }

    public static void main(String[] args) {
        int[] arr = {2, 5, 1, 8, 3};
        System.out.println("Maximum element: " + findMax(arr, arr.length));
    }
}
```

3. Check if an Array is Sorted (Strictly Increasing) using Recursion

```
class CheckSorted {
    static boolean isSorted(int[] arr, int index) {
        if (index == arr.length - 1) return true;
        if (arr[index] >= arr[index + 1]) return false;
        return isSorted(arr, index + 1);
    }

    public static void main(String[] args) {
        int[] arr = {1, 2, 3, 4, 5};
        System.out.println("Is array sorted: " + isSorted(arr, 0));
    }
}
```

4. Check if a String contains only Digits using Recursion

```
class CheckOnlyDigits {
    static boolean isAllDigits(String str, int index) {
        if (index == str.length()) return true;
        if (!Character.isDigit(str.charAt(index))) return false;
        return isAllDigits(str, index + 1);
    }

    public static void main(String[] args) {
        String s = "12345";
        System.out.println("Contains only digits: " + isAllDigits(s, 0));
    }
}
```

5. Count the Number of Zeros in a Number using Recursion

```
class CountZeros {
    static int countZeros(int n) {
        if (n == 0) return 0;
        int last = n % 10;
        return (last == 0 ? 1 : 0) + countZeros(n / 10);
    }

    public static void main(String[] args) {
        int num = 102030;
        System.out.println("Number of zeros: " + countZeros(num));
    }
}
```

6. Convert a Decimal Number to Binary using Recursion

```
class DecimalToBinary {
    static String toBinary(int n) {
        if (n == 0) return "";
        return toBinary(n / 2) + (n % 2);
    }

    public static void main(String[] args) {
        int num = 10;
        String binary = toBinary(num);
        System.out.println("Binary: " + (binary.isEmpty() ? "0" : binary));
    }
}
```

7. Reverse the Digits of a Number using Recursion

```
class ReverseNumber {
    static int reverseHelper(int n, int rev) {
        if (n == 0) return rev;
        return reverseHelper(n / 10, rev * 10 + n % 10);
    }

    static int reverse(int n) {
        return reverseHelper(n, 0);
    }

    public static void main(String[] args) {
        int num = 1234;
        System.out.println("Reversed number: " + reverse(num));
    }
}
```

8. Reverse a Linked List using Recursion

```
class ReverseLinkedList {
    static class Node {
        int data;
        Node next;
        Node(int d) { data = d; }
    }

    static Node reverse(Node head) {
        if (head == null || head.next == null) return head;
        Node newHead = reverse(head.next);
        head.next.next = head;
```

```
        head.next = null;
        return newHead;
    }

    static void printList(Node head) {
        while (head != null) {
            System.out.print(head.data + " ");
            head = head.next;
        }
    }

    public static void main(String[] args) {
        Node head = new Node(1);
        head.next = new Node(2);
        head.next.next = new Node(3);
        head.next.next.next = new Node(4);
        head = reverse(head);
        System.out.print("Reversed Linked List: ");
        printList(head);
    }
}
```

9. Reverse an Array using Recursion

```
class ReverseArray {
    static void reverse(int[] arr, int start, int end) {
        if (start >= end) return;
        int temp = arr[start];
        arr[start] = arr[end];
        arr[end] = temp;
        reverse(arr, start + 1, end - 1);
    }

    public static void main(String[] args) {
        int[] arr = {1, 2, 3, 4, 5};
        reverse(arr, 0, arr.length - 1);
        System.out.print("Reversed Array: ");
        for (int i : arr) System.out.print(i + " ");
    }
}
```

10. Merge Two Sorted Lists using Recursion

```
class MergeTwoSortedLists {
    static class Node {
        int data;
        Node next;
        Node(int d) { data = d; }
    }

    static Node merge(Node l1, Node l2) {
        if (l1 == null) return l2;
        if (l2 == null) return l1;

        if (l1.data < l2.data) {
            l1.next = merge(l1.next, l2);
            return l1;
        } else {
            l2.next = merge(l1, l2.next);
            return l2;
        }
    }

    static void printList(Node head) {
        while (head != null) {
            System.out.print(head.data + " ");
            head = head.next;
        }
    }
    public static void main(String[] args) {
```

```
        Node a = new Node(1);
        a.next = new Node(3);
        a.next.next = new Node(5);

        Node b = new Node(2);
        b.next = new Node(4);
        b.next.next = new Node(6);

        Node merged = merge(a, b);
        System.out.print("Merged List: ");
        printList(merged);
    }
}
```