Q86–Q100: Advanced Recursion and Nested Logic

Q86. Recursive sum of digits

```
public class Main {
    public static int sumDigits(int n) {
        if (n == 0) return 0;
        return (n % 10) + sumDigits(n / 10);
    }

    public static void main(String[] args) {
        System.out.println(sumDigits(1234));
    }
}
```

Q87. Print pattern using recursion

```
public class Main {
    public static void pattern(int n) {
        if (n == 0) return;
        pattern(n - 1);
        for (int i = 0; i < n; i++)
System.out.print("*");
        System.out.println();
    }

    public static void main(String[] args) {
        pattern(4);
    }
}
```

Q88. Fibonacci with memoization

```
import .util.*;

public class Main {
    static Map<Integer, Integer> memo = new
HashMap<>();

    public static int fib(int n) {
        if (n <= 1) return n;
        if (memo.containsKey(n)) return memo.get(n);
        int result = fib(n - 1) + fib(n - 2);
        memo.put(n, result);
        return result;
    }

    public static void main(String[] args) {
        System.out.println(fib(10));
    }
}
```

Q89. Recursive power function

```
public class Main {

    public static int power(int base, int exp) {
        if (exp == 0) return 1;
        return base * power(base, exp - 1);
    }

    public static void main(String[] args) {
        System.out.println(power(2, 5));
    }
}
```

Q90. Count vowels recursively

```
public class Main {
    public static int countVowels(String s, int i) {
        if (i == s.length()) return 0;
        char c = Character.toLowerCase(s.charAt(i));
        if ("aeiou".indexOf(c) != -1)
            return 1 + countVowels(s, i + 1);
        return countVowels(s, i + 1);
    }

    public static void main(String[] args) {
        System.out.println(countVowels("Recursion",
0));
    }
}
```

Q91. Is palindrome (recursive)

```
public class Main {
    public static boolean isPalindrome(String s, int
l, int r) {
        if (l >= r) return true;
        if (s.charAt(l) != s.charAt(r)) return false;
        return isPalindrome(s, l + 1, r - 1);
    }

    public static void main(String[] args) {
        String str = "racecar";
        System.out.println(isPalindrome(str, 0,
str.length() - 1));
    }
}
```

Q92. Tower of Hanoi (Steps)

```
public class Main {
    public static void hanoi(int n, char from, char
to, char aux) {
        if (n == 0) return;
        hanoi(n - 1, from, aux, to);
        System.out.println("Move disk " + n + " from
" + from + " to " + to);
        hanoi(n - 1, aux, to, from);
    }

    public static void main(String[] args) {
        hanoi(3, 'A', 'C', 'B');
    }
}
```

Q93. Nested recursion (McCarthy 91 function)

```
public class Main {
    public static int mcCarthy91(int n) {
        if (n > 100) return n - 10;
        return mcCarthy91(mcCarthy91(n + 11));
    }

    public static void main(String[] args) {
        System.out.println(mcCarthy91(90));
    }
}
```

Q94. Product of array elements

```
public class Main {
    public static int product(int[] arr, int i) {
        if (i == arr.length) return 1;
        return arr[i] * product(arr, i + 1);
    }

    public static void main(String[] args) {
        int[] arr = {2, 3, 4};
        System.out.println(product(arr, 0));
    }
}
```

Q95. Reverse a number using recursion

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

    public static void main(String[] args) {
        System.out.println(reverse(1234, 0));
    }
}
```

Q96. Check if array is sorted

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

    public static void main(String[] args) {
        int[] a = {1, 2, 3, 4};
        System.out.println(isSorted(a, 0));
    }
}
```

Q97. Count consonants in a string

```
public class Main {
    public static int countConsonants(String s, int
i) {
        if (i == s.length()) return 0;
        char c = Character.toLowerCase(s.charAt(i));
        if ("aeiou".indexOf(c) == -1 &&
Character.isLetter(c))
            return 1 + countConsonants(s, i + 1);
        return countConsonants(s, i + 1);
    }

    public static void main(String[] args) {

System.out.println(countConsonants("Recursion", 0));
    }
}
```

Q98. Find minimum in array (recursively)

```
public class Main {
    public static int findMin(int[] arr, int i, int
min) {
        if (i == arr.length) return min;
        if (arr[i] < min) min = arr[i];
        return findMin(arr, i + 1, min);
    }

    public static void main(String[] args) {
        int[] a = {9, 5, 3, 7, 2};
        System.out.println(findMin(a, 0, a[0]));
    }
}
```

Q99. Check power of 2

```java
public class Main {
    public static boolean isPowerOfTwo(int n) {
        if (n == 1) return true;
        if (n % 2 != 0 || n == 0) return false;
        return isPowerOfTwo(n / 2);
    }

    public static void main(String[] args) {
        System.out.println(isPowerOfTwo(16));
    }
}
```

Q100. Print numbers in reverse order

```java
public class Main {
    public static void printReverse(int n) {
        if (n == 0) return;
        System.out.print(n + " ");
        printReverse(n - 1);
    }

    public static void main(String[] args) {
        printReverse(5);
    }
}
```