Q51–Q65: Mathematical and Logical Patterns via Functions (Only)

Q51. Check if a number is prime

```
public class Main {
    public static boolean isPrime(int n) {
        if (n <= 1) return false;
        for (int i = 2; i <= Math.sqrt(n); i++)
            if (n % i == 0) return false;
        return true;
    }

    public static void main(String[] args) {
        System.out.println(isPrime(17));
    }
}
```

Q52. Compute GCD using recursion

```
public class Main {
    public static int gcd(int a, int b) {
        if (b == 0) return a;
        return gcd(b, a % b);
    }

    public static void main(String[] args) {
        System.out.println(gcd(12, 18));
    }
}
```

Q53. Find LCM using GCD

```
public class Main {
    public static int gcd(int a, int b) {
        if (b == 0) return a;
        return gcd(b, a % b);
    }

    public static int lcm(int a, int b) {
        return (a * b) / gcd(a, b);
    }
```

```
    public static void main(String[] args) {
        System.out.println(lcm(12, 18));
    }
}
```

Q54. Sum of digits using recursion

```
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));
    }
}
```

Q55. Reverse digits recursively

```
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(123, 0));
    }
}
```

Q56. Check if number is Armstrong

```
public class Main {
    public static int power(int base, int exp) {
        return (int) Math.pow(base, exp);
    }

    public static boolean isArmstrong(int num) {
        int sum = 0, temp = num, digits =
String.valueOf(num).length();
        while (temp > 0) {
            int digit = temp % 10;
```

```
            sum += power(digit, digits);
            temp /= 10;
        }
        return sum == num;
    }

    public static void main(String[] args) {
        System.out.println(isArmstrong(153));
    }
}
```

Q57. Count number of digits

```
public class Main {
    public static int countDigits(int n) {
        if (n == 0) return 0;
        return 1 + countDigits(n / 10);
    }

    public static void main(String[] args) {
        System.out.println(countDigits(10234));
    }
}
```

Q58. Convert decimal to binary (recursive)

```
public class Main {
    public static void toBinary(int n) {
        if (n == 0) return;
        toBinary(n / 2);
        System.out.print(n % 2);
    }

    public static void main(String[] args) {
        toBinary(10);
    }
}
```

Q59. Find sum of first N natural numbers

```
public class Main {

    public static int sumN(int n) {
        if (n == 1) return 1;
        return n + sumN(n - 1);
    }

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

Q60. Check if a number is palindrome

```
public class Main {
    public static boolean isPalindrome(int n) {
        return n == reverse(n, 0);
    }

    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(isPalindrome(121));
    }
}
```

Q61. Print factorial series up to N

```
public class Main {
    public static int factorial(int n) {
        if (n <= 1) return 1;
        return n * factorial(n - 1);
    }

    public static void main(String[] args) {
        for (int i = 1; i <= 5; i++)
            System.out.print(factorial(i) + " ");
    }
}
```

Q62. Count even digits in a number

```
public class Main {
    public static int countEven(int n) {
        if (n == 0) return 0;
        int digit = n % 10;
        return (digit % 2 == 0 ? 1 : 0) + countEven(n
/ 10);
    }

    public static void main(String[] args) {
        System.out.println(countEven(20468));
    }
}
```

Q63. Print N-th Fibonacci number (recursive)

```
public class Main {
    public static int fib(int n) {
        if (n <= 1) return n;
        return fib(n - 1) + fib(n - 2);
    }

    public static void main(String[] args) {
        System.out.println(fib(6));
    }
}
```

Q64. Check if number is strong (sum of digit factorials = number)

```
public class Main {
    public static int factorial(int n) {
        if (n <= 1) return 1;
        return n * factorial(n - 1);
    }

    public static boolean isStrong(int num) {
        int sum = 0, temp = num;
        while (temp > 0) {
            int digit = temp % 10;
            sum += factorial(digit);
            temp /= 10;
        }
        return sum == num;
    }
```

```
    public static void main(String[] args) {
        System.out.println(isStrong(145));
    }
}
```

Q65. Count digits that divide the number itself

```
public class Main {

    public static int countDivisibleDigits(int num) {
        int count = 0, temp = num;
        while (temp > 0) {
            int digit = temp % 10;
            if (digit != 0 && num % digit == 0)
                count++;
            temp /= 10;
        }
        return count;
    }

    public static void main(String[] args) {

System.out.println(countDivisibleDigits(120));
    }
}
```

.