Q66–Q75: Functional Decomposition & Modular Code

Q66. Check if a string is a palindrome using helper functions

```
public class Main {
    public static boolean isPalindrome(String s) {
        s = preprocess(s);
        return s.equals(reverse(s));
    }

    public static String preprocess(String s) {
        return s.toLowerCase().replaceAll("[^a-z]",
"");
    }

    public static String reverse(String s) {
        return new
StringBuilder(s).reverse().toString();
    }

    public static void main(String[] args) {
        System.out.println(isPalindrome("Madam"));
    }
}
```

Q67. Validate a password using multiple helper checks

```
public class Main {
    public static boolean isValidPassword(String
password) {
        return hasUpper(password) &&
hasDigit(password) && password.length() >= 8;
    }

    public static boolean hasUpper(String s) {
        for (char c : s.toCharArray())
            if (Character.isUpperCase(c)) return
true;
        return false;
    }

    public static boolean hasDigit(String s) {
        for (char c : s.toCharArray())
```

```
            if (Character.isDigit(c)) return true;
        return false;
    }

    public static void main(String[] args) {

System.out.println(isValidPassword("Hello123"));
    }
}
```

Q68. Find all prime numbers in a range using modular functions

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

    public static void printPrimesInRange(int start,
int end) {
        for (int i = start; i <= end; i++)
            if (isPrime(i)) System.out.print(i + "
");
    }

    public static void main(String[] args) {
        printPrimesInRange(10, 30);
    }
}
```

Q69. Break number into digits and print even ones

```
public class Main {
    public static void printEvenDigits(int num) {
        while (num > 0) {
            int digit = num % 10;
            if (isEven(digit)) System.out.print(digit
+ " ");
```

```
            num /= 10;
        }
    }

    public static boolean isEven(int n) {
        return n % 2 == 0;
    }

    public static void main(String[] args) {
        printEvenDigits(4821);
    }
}
```

Q70. Sort three numbers using helper swap and compare functions

```
public class Main {
    public static void sort(int a, int b, int c) {
        int[] arr = {a, b, c};
        for (int i = 0; i < arr.length - 1; i++)
            for (int j = i + 1; j < arr.length; j++)
                if (arr[i] > arr[j]) swap(arr, i, j);

        for (int num : arr) System.out.print(num + " 
");
    }

    public static void swap(int[] arr, int i, int j) 
{
        int t = arr[i];
        arr[i] = arr[j];
        arr[j] = t;
    }

    public static void main(String[] args) {
        sort(42, 17, 9);
    }
}
```

Q71. Count vowels and consonants using modular design

```
public class Main {
    public static boolean isVowel(char c) {
        return "aeiouAEIOU".indexOf(c) != -1;
    }
    public static void countVowelsConsonants(String
s) {
        int vowels = 0, consonants = 0;
        for (char c : s.toCharArray()) {
            if (Character.isLetter(c)) {
                if (isVowel(c)) vowels++;
                else consonants++;
            }
        }
        System.out.println("Vowels: " + vowels + ",
Consonants: " + consonants);
    }

    public static void main(String[] args) {
        countVowelsConsonants("Functional
Decomposition");
    }
}
```

Q72. Validate if a year is leap year using helper

```
public class Main {
    public static boolean isLeapYear(int year) {
        return (year % 4 == 0 && year % 100 != 0) ||
(year % 400 == 0);
    }

    public static void main(String[] args) {
        System.out.println(isLeapYear(2024));
    }
}
```

Q73. Convert temperature between Celsius and Fahrenheit

```
public class Main {
    public static double toFahrenheit(double celsius)
{
        return (celsius * 9/5) + 32;
    }

    public static double toCelsius(double fahrenheit)
{
        return (fahrenheit - 32) * 5/9;
    }

    public static void main(String[] args) {
        System.out.println("37°C in F = " +
toFahrenheit(37));
        System.out.println("98.6°F in C = " +
toCelsius(98.6));
    }
}
```

Q74. Calculate compound interest using decomposition

```
public class Main {
    public static double compoundInterest(double p,
double r, int t) {
        return p * Math.pow(1 + r / 100, t);
    }

    public static void main(String[] args) {
        System.out.println(compoundInterest(1000, 5,
2));
    }
}
```

Q75. Generate multiplication table using functions

```
public class Main {
    public static void printTable(int n) {
        for (int i = 1; i <= 10; i++)
            printRow(n, i);
    }

    public static void printRow(int n, int
multiplier) {
        System.out.println(n + " x " + multiplier + "
= " + (n * multiplier));
    }

    public static void main(String[] args) {
        printTable(7);
    }
}
```