Quiz 2 (5.5 points) - V1

Date: July 16, 2025
Time limit: 45 minutes
Name: **Student Number**:

The Greatest Common Divisor (GCD) of two integers is the largest integer that can exactly divide both numbers without leaving a remainder.

- $GCD(6, 9) = 3$
- $GCD(48, 18) = 6$
- $GCD(101, 103) = 1$

Write a C program that calculates the GCD of two integers.

1. Implement a `gcd()` function (3 points):
   - Takes two integers as parameters.
   - Computes and returns their GCD.
2. Implement in `main()` (2.5 points):
   - Declare two integer variables and initialize them with test values.
   - Call the `gcd()` function with these values, and store the returned result in a variable.
   - Print the result to the terminal in the format:

     ```
     GCD of a and b is: result
     ```

     Example Output:

     ```
     GCD of 48 and 18 is: 6
     ```

Constraints:

- The function must work for any pair of positive integers.
- You can assume that the input numbers are non-zero.

Quiz 2 (5.5 points) - V2

Date: July 16, 2025
Time limit: 45 minutes
Name: **Student Number**:

The factorial of a non-negative integer nn is defined as:

$$n! = n \times (n-1) \times (n-2) \times ... \times 1$$

For example:

- $5! = 5 \times 4 \times 3 \times 2 \times 1 = 120$
- $0! = 1$ (by definition)

Write a C program to compute the factorial of a given number.

1. Implement a `factorial()` function (3 points):
   - To compute factorial of an input value. You can use a `for` loop.
   - Returns the factorial.
2. Implement in `main()` (2.5 points):
   - Declare an integer variable `n` and initialize it with a test value.
   - Call `factorial()` function with `n` as an argument and store the returned result in a variable.
   - Print the result to the terminal in the format:

     ```
     Factorial of n is: result
     ```

     Example Output:

     ```
     Factorial of 5 is: 120
     ```

Constraints:

- Assume $n \geq 1$
- Factorial values can grow large, but for this problem, assume $n \leq 12$.

Quiz 2 (5.5 points) - V3

Date: July 16, 2025
Time limit: 45 minutes
Name: **Student Number**:

A prime number is a natural number greater than 1 that has no divisors other than 1 and itself. For example:

- Prime numbers: 2, 3, 5, 7, 11, 13, 17, ...
- Non-prime numbers: 4, 6, 8, 9, 10, ... (because they have divisors other than 1 and themselves)

Write a C program that checks whether a given number is prime or not.

1. Implement a `isPrime` function (3 points):
   - Takes an integer as input.
   - Returns `true` (or `1`) if the number is prime, `false` (or `0`) otherwise.
2. Implement in `main()` (2.5 points):
   - Declare an integer variable and initialize it with a test value.
   - Call the `isPrime()` function with the test value, and store the returned result in a variable.
   - Print an appropriate message based on the result:

     ```
     n is a prime number.
     ```

     Example Output:

     ```
     7 is a prime number.
     ```

     or

     ```
     n is not a prime number.
     ```

     Example Output:

     ```
     10 is not a prime number.
     ```

Constraints:

- The function must work for any integer $n \geq 1$.
- You may assume n is positive.
- Efficiency is not a primary concern; the focus is on ensuring the code functions correctly.

Quiz 2 (5.5 points) - V4

Date: July 16, 2025
Time limit: 45 minutes
Name: **Student Number**:

Given a 1D array of numbers, the average (mean) is computed as:

$$Average = \frac{\sum elements\ in\ array}{number\ of\ elements}$$

Write a C program that calculates the average of the elements in a 1D array.

1. Implement a `aveArray()` function (3 points):
   - Takes a 1D array of floating point values and its size as parameters.
   - Computes the average and returns the computed average.
2. Implement in `main()` (2.5 points):
   - Define and initialize the following array:

     $$array = [-3,\ \ 2.36,\ \ 10^{-3},\ \ 1234.567,\ \ -56]$$

   - Call the `aveArray()` function and store the returned result in a variable.
   - Print the result to the terminal in the format:

     ```
     Average of the array is: result
     ```

     Example Output:

     ```
     Average of the array is: 2.355856e+02
     ```

Constraints:

- All values in `array` are represented with up to 6 decimal places in floating-point precision.
- The function must be able to handle arrays of different sizes without modification.
- The result should be printed with a reasonable number of decimal places.