Factorial Digit Sum Implementation

Multiple Programming Languages

📜 Definition — Factorial Digit Sum

A **Factorial Digit Sum Number** (also known as a **Factorion**) is a number that is equal to the sum of the factorials of its digits.

Examples:

- 145 = 1! + 4! + 5! = 1 + 24 + 120 = 145 ✅
- 2 = 2! = 2 ✅
- 123 = 1! + 2! + 3! = 1 + 2 + 6 = 9 ❌

📝 Problem

Given an integer n, determine whether it is a Factorial Digit Sum Number. If it is, output `1`. Otherwise, output `0`.

📥 Input

- A single integer n ($0 \le n \le 10^6$)

📤 Output

- Output `1` if n is a Factorial Digit Sum Number.
- Output `0` otherwise.

💡 Example

Input:

```
145
```

Output:

```
1
```

Explanation:

- Digits: 1, 4, 5

- 1! = 1, 4! = 24, 5! = 120
- Sum = 145 → matches the original number → print `1`

Algorithm

1. Take an integer input `n`
2. For each digit in the number, calculate its factorial
3. Sum all the factorials
4. Output "1" if the sum equals the original number, "0" otherwise

---

C Implementation

```c
#include <stdio.h>
int main() {
    int i, n, s = 0, j;
    scanf("%d", &n);
    for (j = n; j > 0; j = j / 10) {
        int t = 1;
        for (i = 1; i <= j % 10; i++) {
            t = t * i;
        }
        s = s + t;
    }
    if (s == n)
        printf("1");
    else
        printf("0");
    return 0;
}
```

C++ Implementation

```cpp
```

```
#include <iostream>
using namespace std;

int main() {
    int i, n, s = 0, j;
    cin >> n;
    for (j = n; j > 0; j = j / 10) {
        int t = 1;
        for (i = 1; i <= j % 10; i++) {
            t = t * i;
        }
        s = s + t;
    }
    if (s == n)
        cout << "1";
    else
        cout << "0";
    return 0;
}
```

Rust Implementation

```
rust
```

```
use std::io;

fn main() {
    let mut input = String::new();
    io::stdin().read_line(&mut input).expect("Failed to read line");
    let n: i32 = input.trim().parse().expect("Invalid number");

    let mut s = 0;
    let mut j = n;

    while j > 0 {
        let digit = j % 10;
        let mut t = 1;

        for i in 1..=digit {
            t *= i;
        }

        s += t;
        j /= 10;
    }

    if s == n {
        println!("1");
    } else {
        println!("0");
    }
}
```

C# Implementation

```
csharp
```

```
using System;

class Program {
    static void Main() {
        int n = int.Parse(Console.ReadLine());
        int s = 0;
        int j = n;

        while (j > 0) {
            int digit = j % 10;
            int t = 1;

            for (int i = 1; i <= digit; i++) {
                t *= i;
            }

            s += t;
            j /= 10;
        }

        if (s == n) {
            Console.WriteLine("1");
        } else {
            Console.WriteLine("0");
        }
    }
}
```

Language-Specific Notes

C

- Uses `stdio.h` for input/output functions
- Original implementation with basic C syntax

C++

- Uses `iostream` for input/output instead of `stdio.h`
- Leverages `cin` and `cout` for cleaner I/O operations

Rust

- Uses Rust's memory-safe syntax with explicit error handling
- Employs `std::io` for input operations
- Range syntax `1..=digit` for inclusive loops

C#

- Uses .NET's Console class for I/O operations
- Follows C# naming conventions and syntax
- Leverages automatic memory management

Example Usage

Test Case 1: Input: 145

Calculation: 1! + 4! + 5! = 1 + 24 + 120 = 145

Output: 1

Test Case 2: Input: 123

Calculation: 1! + 2! + 3! = 1 + 2 + 6 = 9 ≠ 123

Output: 0

Test Case 3: Input: 2

Calculation: 2! = 2

Output: 1