Written Exercise for Lab06 recursion

Name: Rong Huang

Date:03/10/2024

1. What data type in the C programming language allows for the largest values of factorial to be computed?

In C, the unsigned long long int data type allows for the largest values of factorial to be computed due to its ability to store the largest range of positive integers.

Specifically, it can represent numbers from 0 up to $2^{64}-1$ (which is 18,446,744,073,709,551,615). This is significantly larger than what int (typically $2^{31}-1$ for signed int, or $2^{32}-1$ for unsigned int) and long int (typically $2^{63}-1$ for signed long, or $2^{64}-1$ for unsigned long) can represent. The choice of unsigned long long int over unsigned int or unsigned long int is due to this larger range, making it more suitable for computing and representing large factorials without overflow.

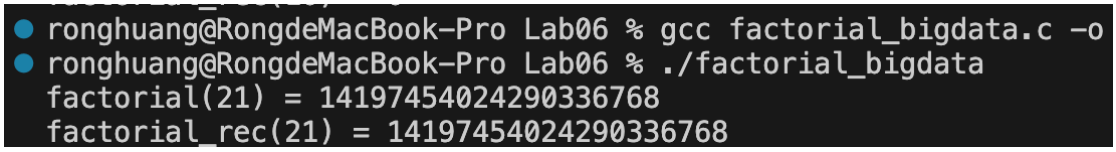
(According to the website: https://www.geeksforgeeks.org/data-types-in-c/)

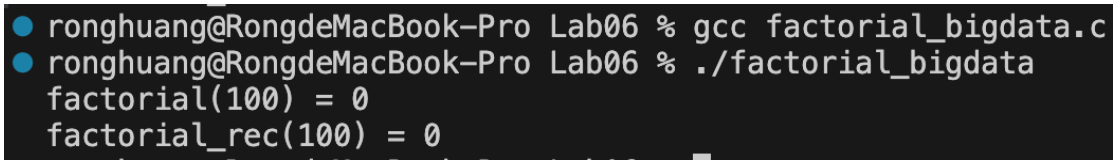
2. At what input value for the recursive factorial function does your computer start to 'crash' or really slow down when you try to compute a factorial? Is it the same value as the iterative function? Experiment and report your result.

When the input value reaches 21, both the iterative and recursive functions yield an incorrect result of 14,197,454,024,290,336,768, differing from the correct factorial value of 51,090,942,171,709,440,000. This occurs because the computed value exceeds the maximum range that unsigned long long int can represent, which is from 0 to 18,446,744,073,709,551,615, leading to an overflow.

Interestingly, both functions encounter this overflow issue at the same input value, demonstrating that the limitation is due to the data type's capacity rather than the specific algorithmic approach. For inputs significantly larger, such as 100, the output is 0 for both methods. This outcome is a result of multiple overflows during the calculation process, which is a characteristic of the data type's handling of operations exceeding its representable range.

Thus, while both iterative and recursive implementations show similar behavior in terms of overflow, their limitations in handling large factorials are primarily governed by the unsigned long long int data type's maximum value.





3. In 2-3 sentences, describe why you believe you saw or didn't see differences between the iterative and recursive versions of factorial.

In my experiment, no significant differences were observed between the iterative and recursive versions of the factorial function in terms of their ability to compute large factorials up to the point of data type limitations since both implementations encountered similar limitations due to the unsigned long long int data type's maximum value, leading to overflow at the same input value.

However, the iterative method proves more efficient for large numbers due to its minimal resource consumption. Unlike recursion, which can lead to stack overflow for very large inputs due to the depth of recursive calls, the iterative approach avoids this by maintaining a single execution context, making it more suited for computing large factorials.