Task 3- Combined Reflection

Question 1: What numerical property limits integer types

Integer types are limited by their finite representable range, which depends on the number of bits used to store them. Each integer type (for example, 1, 2, 4, or 8 bytes) can only represent values between a specific minimum and maximum, defined as:

$$-2^{n-1} \leq \text{integer} \leq 2^{n-1} - 1$$

where *n* is the number of bits available for the integer representation.
When a calculation produces a value outside this range, overflow occurs, causing incorrect or unpredictable results.

Question 2: What numerical property limits real types?

real types are limited by their finite precision and finite exponent range, both determined by the number of bits used for their representation.

A floating-point number can represent only a certain number of significant digits and magnitudes within a specific exponent range.
When numbers become too small to affect larger values, this limit is reached, which is known as the machine epsilon.
As a result, operations may lose accuracy due to loss of significance when combining very small and large values.

Question 3: Which error (overflow or precision loss) is more dangerous for scientific simulations, and why?

Precision loss is more dangerous for scientific simulations because it can change results without being noticed, while overflow usually causes clear errors.

For example:

- Overflow (factorial):
  When calculating a large factorial, like n!, the number quickly becomes too big for the computer to store. Once it exceeds the limit, it wraps around or becomes negative, which is an obvious overflow error that can be detected.
- Precision loss (sum):

Precision loss is more dangerous for scientific simulations because it changes results silently, while overflow produces an obvious error.

As an example for the integer type (4) in Task 1, the factorial value grows so large that it exceeds the representable range. When this happens, the true value (for instance, about 6.2×10^9) cannot be stored, and the result suddenly becomes incorrect or even negative, which is a clear sign of overflow.

However, in Task 2, when repeatedly adding smaller numbers to 1.0 in double precision, the additions stop changing the result once the added value becomes about 1×10^{-16}. This shows precision loss because the computer can no longer represent differences smaller than the machine epsilon.

Therefore, precision loss is more dangerous because it occurs gradually and unnoticed, while overflow is abrupt and easier to detect.