(If I detect AI, I will punish you) Homework 1:

1) **Implement functions for both single and double precision floating point:**
   a) Create function to convert floating-point value to its bitwise representation.
   b) Create function to convert bitwise representation of floating-point value to an actual floating-point value.

2) **Implement functions for both single and double precision floating point:**
   a) Check if floating-point value is **finite** value. Refer to **isfinite** function in C++ language.
   b) Check if floating-point value is **any infinity** value. Refer to **isinf** function in C++ language.
   c) Check if floating-point value is **positive infinity** value.
   d) Check if floating-point value is **negative infinity** value.
   e) Check if floating-point value is **any zero** value.
   f) Check if floating-point value is **positive zero** value.
   g) Check if floating-point value is **negative zero** value.
   h) Check if floating-point value is **not a number** value.
   i) Check if floating-point value is **normal** value**.**
   j) Check if floating-point value is **subnormal** value**.**
   k) Check if floating-point value is signed. Refer to **signbit** function in C++ language.
   l) Classify floating-point value. See **fpclassify** function in C++ language.

3) **Implement functions for both single and double precision floating point:**
   a) To get an **absolute** value of floating-point value. Refer to **abs** function in C++ language.
   b) To get **min** of two values of floating-point value. Refer to **min** function in C++ language.
   c) To get **max** of two values of floating-point value. Refer to **max** function in C++ language.
   d) To **clamp** between two floating-point values. Refer to clamp function in C++ language.

4) **Implement functions for both single and double precision floating point:**
   a) Compare two floating-point values for **equality** with specified precision.
   b) Compare two arbitrary floating-point values for **equality**.
   c) Compare two floating-point values for through **less** operator with specified precision.
   d) Compare two arbitrary floating-point values through **less** operator.
   e) Compare two floating-point values for **greater** operator with specified precision.
   f) Compare two arbitrary floating-point values through **greater** operator.

5) **Cover implemented functions through unit-testing with framework like Google Tests or Boost Tests**
6) **Compute decimal representation of:**
   a) Compute decimal representation of **00111111000110011001100110011010** *(Little Endian)*
   b) Compute decimal representation of **01000010111101110011100110010011** *(Little Endian)*
   c) Compute decimal representation of **11000011010111101001110011001010** *(Little Endian)*

7) **Provide a detailed response to the questions below:**
   d) We have a floating-point variable **X**. Is it possible for expression (**X != X**) to be true?
   e) We have a floating-point variable **X**, we have expression (**Y = X * 0**). Is it safe to assume that **Y** is always **0**?
   f) We have a floating-point variables **X** and **Y**. Is it possible for expression (**(X + Y) == X**) to be true?
   g) We have a floating-point variables **X**, **Y** and **Z**. It is safe to assume that (**(X + Y) + Z)** == **(X + (Y + Z)**) is true?

For extra points:

- Implement **ceil**, **trunc,** and **floor** functions using only functions from the **first** and **second** block. Function must be covered with unit-test. Function must work correctly with all finite numbers that can be stored inside the single or double precision floating point.