# C: Understanding IEEE 754 Floating-Point Representation

CSCI210: Computer Organization - Professor Tolley

Due Date: 03-15-24

## Objective

This assignment is designed to deepen your understanding of floating-point number representation in C, specifically the IEEE 754 standard. You will work with a provided codebase that includes functions for representing, manipulating, and converting floating-point numbers at the bit level.

## Understanding Structs, Unions, and Bitfields

Before diving into the assignment, it is important to understand the concepts of structs, unions, and bitfields in C programming.

**Structs** are used to group variables of different types under a single name. Here is an example of a struct in C:

```
typedef struct {
    int id;
    char name[50];
    float salary;
} Employee;
```

**Unions** allow different data types to share the same memory location. Here is an example of a union in C:

```
typedef union {
    int id;
    float salary;
    char name[50];
} Data;
```

**Bitfields** in structs allow the allocation of a specific number of bits for a field. They are useful for memory-efficient storage. Here is an example of a struct with bitfields:

```
typedef struct {
    unsigned int sign: 1;
    unsigned int exponent: 8;
    unsigned int mantissa: 23;
} FloatRepresentation;
```

In this assignment, you will use a union to interpret the bit representation of a floating-point number both as a `float` and as a struct with individual fields for the sign, exponent, and mantissa.

## Tasks

Your assignment involves the following tasks:

1. **Implement the calculateMantissa Function**: Write the function to calculate the mantissa from its bit-level representation. This exercise helps understand how the fractional part of a floating-point number is stored and interpreted.
2. **Implement the calculateExponent Function**: Implement the function to calculate the exponent from its bit-level representation, focusing on understanding how the exponent is stored in a biased format.
3. **Implement the isNormalized Function**: Determine if a given floating-point number is a normalized number according to the IEEE 754 standard, learning about the range of normalized floating-point numbers.
4. **Implement the toDecimal Function**: Complete this function to convert the bit-level representation of a floating-point number back to its decimal value, combining understanding of both the mantissa and exponent.
5. **Enhance the Main Function** (Optional): Enhance the main function to include additional features, such as handling special cases (like infinity or NaN), or improving user interaction.

## Submission Guidelines

- Submit the modified C source file.
- Include a separate document with a brief explanation of your code and the changes you made.
- Ensure your code is well-commented and follows standard coding conventions.

## Evaluation Criteria

- Correctness and efficiency of the modifications and extensions.
- Clarity and completeness of your documentation and comments.
- Adherence to coding standards and best practices.

## Resources

To complete this assignment, you may refer to the following:

- IEEE 754 Standard Documentation
- C Programming Language References in Files on Canvas
- Course textbooks and lecture notes

Good luck, and feel free to reach out if you have any questions!