Files to submit: MyFloat.cpp, MyFloat.h
Time it took Matthew to complete: 1 hour

- All programs must compile without warnings when using the -Wall and -Werror options
- Submit only the files requested
  - Do **NOT** submit folders or compressed files such as .zip, .rar, .tar, .targz, etc
- If submitting in a group on Grade Scope please make sure to mark your partner.
  - Only one of you has to submit there
- Your program must match the output exactly to receive credit.
  - Make sure that all prompts and output match mine exactly.
  - Easiest way to do this is to copy and paste them
- All input will be valid unless stated otherwise
- Print all real numbers to two decimal places unless otherwise stated
- The examples provided in the prompts do not represent all possible input you can receive.
- All inputs in the examples in the prompt are underlined
  - You don't have to make anything underlined it is just there to help you differentiate between what you are supposed to print and what is being given to your program
- If you have questions please post them on Piazza

1. For this assignment you will be implementing floating point add and subtract without using the hardware's floating point or double precision add. This will give you a better understanding of how difficult they are to work with and a higher appreciation of the hardware for doing this for you. You will be turning in a file called **MyFloat.cpp** and its associated header file, **MyFloat.h**, that implements your own floating point number. This object should support both + and -.
    1. You may not use floating point or double precision add in your solution. This means that you should not have the following in your program:
        1. float x,y;
        2. x + y;
    2. MyFloat should represent a float using three unsigned integers: sign, exponent, and mantissa.
    3. MyFloat must have the following private methods defined on it. These functions must be **implemented using inline assembly.**
        1. void unpackFloat(float f);
            1. Given a float f this function should set sign, exponent, and mantissa to the appropriate values based on the value of f.
        2. float packFloat() const;
            1. This function should return the floating point representation of MyFloat
        3. You may declare and initialize variables to a constant in C as well as return a value but you should do no other calculations.
    4. MyFloat must have the following public functions defined on it
        1. MyFloat operator+(const MyFloat& rhs) const;
            1. This function should add this to rhs and return the result of the addition
            2. When adding the two numbers, the maximum amount of precision must be maintained.
                1. Before doing the addition you should restore the leading 1. This means that the mantissa will end up taking 24 bits.
                2. Since you are adding two 24 bit numbers together the result could take up to 25 bits.
                3. Be careful when shifting. Since the numbers are 32 bits, the maximum amount

you can shift either left or right is 31. If you try to shift by more than this, nothing happens.
    3. After doing the addition the number should be returned to its normalized form.
        1. When normalizing the number we will truncate it down to the 23 most significant bits.
2. MyFloat operator-(const MyFloat& rhs) const;
    1. This function return this – rhs.
    2. The maximum amount of precision must be maintained
        1. One thing to watch out for when subtracting (or adding numbers with different signs) is that you may need to borrow. A borrow would occur if the most significant bit that is right shifted out is a 1.
    3. The number should be returned to normalized form after adding.
    4. I highly suggest you call + after slightly modifying rhs
3. bool operator==(const float rhs) const;
    1. Returns true if this represents the float on the right hand side.
5. You have been provided with a main.cpp that will read in arguments from the command line and then call your function. Your code must be callable from main.cpp
    1. Arg1 is a floating point number
    2. Arg2 is either + or -
    3. Arg3 is another floating point number
6. You have also been provided with a header file for MyFloat and a partially completed MyFloat.cpp.
    1. Feel free to add additional methods but do not remove any.
7. Finally you have been provided with a makefile to compile your submission. Your submission must be compilable by the given makefile.
8. Your CPU may use a different rounding scheme than what we are using your floating point add/subtract may not match float a + b. You shouldn't try to match the computer but should instead match my answers. If you do think I made a mistake though please let me know.

Examples

```
./fpArithmetic.out 10 +  7
My Add: 17

./fpArithmetic.out .5 +  .5
My Add: 1

./fpArithmetic.out 1736217621 + 0.5
My Add: 1.73622e+09

./fpArithmetic.out -5 + 5
My Add: 0

./fpArithmetic.out 100 - 50
My Subtraction: 50

./fpArithmetic.out 10.3 - 5.1
My Subtraction: 5.2
```