Due: Smartsite Wed., 12/11, 11:55 p.m.
Names of Files to Submit: **MyFloat.cpp, MyFloat.h, ReadMe.txt**

- All prompts for input and all output must match my prompts/output. We use a program to grade your work and tiny differences can cause your work to be marked as a 0.
  - If your work does happen to lose points you have 1 week from receiving your grade to request a regrade and we will be happy to go back and fix any typos in your prompt/output.
  - The best way to avoid being deducted points is to copy the prompt/unchanging portion of the outputs into your code. Make sure to get the spaces as well.
- You must also submit a file called ReadMe.txt. The format is as follows
  - student: first name last name, id
  - student: first name last name, id
  - …
  - notes: Any difficulties you had with the assignment and any comments you would like to make. Also if a program doesn't work please let us know.
- Submit your work on Smartsite by uploading each file separately. **DO NOT** upload any folders or compressed files such as .rar, .tar, .targz, .zip etc.
- **ALL** students in a group should upload the submission to SmartSite.
- If you have any questions please post them to 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 need to 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
   4. MyFloat should also have the following private static function
      1. `static bool carryWouldHappen(unsigned int a, unsigned int b);`
         1. This function should return true if adding a and b together would result in a carry and false otherwise
   5. 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. Since we are storing the mantissa in a 32 bit number it can be expanded up to 32 bits

        2. Before doing the addition you should restore the leading 1. This means that the mantissa will end up taking 24 bits.
        3. Since mantissas can potentially take up 32 bits this means that when they are added together that a carry could occur. This carry should not be discarded but kept track of because we want to preserve the maximum precision.
    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. Remember to take into account if a carry happened
2. MyFloat operator-(const MyFloat& rhs) const;
    1. This function return this – rhs.
    2. The maximum amount of precision must be maintained
    3. The number should be returned to normalized form after adding.
    4. I highly suggest you call + after slightly modifying rhs
6. 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
7. 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.
8. Finally you have been provided with a makefile to compile your submission. Your submission must be compilable by the given makefile.
9. 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.

2. You have been provided with a testing program, called tester.py, that will check if your solution is correct. It requires python3 to run and assumes that your compiled solution, the Tests directory and the Solutions directory will be located in the directory as itself. To run the tester program do
    1. make
    2. python3 tester.py

    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
```