Purpose: This program will involve working with floating point arithmetic and local variables to calculate the hypotenuses for a series of triangles and finding their median hypotenuse.

You will be provided a template file to start your source code. This file includes the data for 2 sides of the triangle (one as floats and the other as an integer) as well as the general algorithm to follow as comments.

A debug input file will be provided to examine the values of the program.

Program Specifications:
You will need to write two functions to complete this program.

1. Find Median Hypotenuse
The first function will take as arguments:
1. The array of side A values by reference (floating points).
2. The array of side B values by reference (integers).
It should return the median hypotenuse value of the triangles (floating point).
In order to find the median, you will need to store the calculated hypotenuses and sort the values (using the second function). Store the values in a local array and send the comb sort function the address.

Hypotenuse (C):

$$C_i = \sqrt{A_i^2 + B_i^2}$$

2. Comb Sort (Float Version)
Convert the comb sort function from assignment #6 to compare floating point values rather than integers.

Expected Output:

Set a breakpoint in the find median function near the end of the function (before popping rbp). Use the provided debugger script to output the contents of the local array and the return value to the output file.

Example answers.txt output by debug script

```
Float Values:
0x7fffffffded4:       1.0562197      2.48805141     2.79501343     3.26955652
0x7fffffffdee4:       4.09140539     4.23236322     5.23473978     5.63568115
0x7fffffffdef4:       5.91692495     5.99096012     6.02408504     6.22274065
0x7fffffffdf04:       6.53118658     6.77383947     7.48801708     8.05084515
0x7fffffffdf14:       8.27212811     8.51765251     8.6838932      8.88119411
0x7fffffffdf24:       8.89607239     9.00555897     9.35016537     9.44047165
0x7fffffffdf34:       9.59629631     9.66910553     9.75213242     9.83569527
0x7fffffffdf44:       10      10.031949      10.3324728     10.5060034
0x7fffffffdf54:       10.7379007     10.7472038     11.0154848     11.0154848
0x7fffffffdf64:       11.0591183     11.2340374     11.3438787     11.6928396
0x7fffffffdf74:       11.767621      11.8769732     11.8780632     11.9293289
0x7fffffffdf84:       12.1240301     12.2015123     12.3725538     12.3956804
0x7fffffffdf94:       12.4588327     12.4871178     12.5612469     12.5826712
0x7fffffffdfa4:       12.7254429     12.8272991     13.0309782     13.1308832
0x7fffffffdfb4:       13.4161444     13.479599      13.7381372     13.9925117
0x7fffffffdfc4:       14.0464373     14.1365337     14.1488132     14.5873365
0x7fffffffdfd4:       14.9848452     15.0434341     15.0768166     15.1083546
0x7fffffffdfe4:       15.143527      15.2958689     15.5988331     15.625905
0x7fffffffdff4:       16.5680294     17.406002      18.1732655

Median:
$1 = {v4_float = {11.2340374, 0, 0, 0}, v2_double = {5.4046343660962038e-315, 0}, v16_int8 = {-98, -
66, 51, 65, 0 <repeats 12 times>}, v8_int16 = {-16738, 16691, 0, 0, 0, 0, 0, 0}, v4_int32 =
{1093910174, 0, 0, 0}, v2_int64 = {1093910174, 0}, uint128 = 1093910174}
```

The first value of v4_float is the one we are interested in for the median.

Submission

Once you are satisfied with the program, upload the assembly source code (.asm) file to the class website.