Physics 2G03 - Homework 5

1. The extra include statement, *#include <cmath>*, tells the compiler to treat the contents of the file *cmath* as if it appears in the source program. In a sense, it copies the code from *cmath* and pastes it into the source file. Thus, the source file can access all of the functions in *cmath*. You can also think of it as importing external code. *cmath* is a standard library header that contains useful math functions. The *include* statements allows us to import it into our program and use the available functions instead of recreating them ourselves. This is one of the benefits of Object Oriented Programming; code re-usability.

2. Please see file: *sine1file.cpp*

3. Please see executable file: *sine1file*

4.

sine1file outputs the following:			
Sine(0.5)	**Sine(1)**	**Sine(2)**	**Sine(4)**
0.479427	0.841667	0.933333	1.86667

sinestandard outputs the following values for sine:			
Sine(0.5)	**Sine(1)**	**Sine(2)**	**Sine(4)**
0.479426	0.841471	0.909297	-0.756802

5. The Taylor series in *sine1file.cpp* is not useful for estimating values greater than ~1.5 or less than ~(-1.5). In other words, the Taylor series in question cannot properly estimate values that are far away from 0. Thus, the Taylor series is not useful at all.

6. I think a smarter way to estimate values of *x* bigger than 1.5 is to use a better Taylor series or equation that decreases monotonically. For example, the equation in homework 4 was decent for estimating the values of *pi*; it was accurate to 7-8 decimal places. A better equation for estimating *sine* could be:
((-1)^n) * ((x^(2*n+1)) / (2*n+1)!)
It's a little complicated, but once implemented, it will be more accurate than the Taylor series above.

7. Please see *sine.cpp* and *sinemain.cpp*

8. Please see the executable file called *sine*

9.

```
(1) sine: sinemain.o sine.o
(2)     c++ sinemain.o sine.o -o sine
(3) sinemain.o: sinemain.cpp sine.h
(4)     c++ sinemain.cpp -c
(5) sine.o: sine.cpp sine.h
(6)     c++ sine.cpp -c
```

The 1st line contains information about what is needed to make the executable *sine*. It needs the object files, *sinemain.o* and *sine.o*.
The 2nd line is the action that will produce the executable s*ine*. The compiler needs to link the object files, *sinemain.o* and *sine.o*, to create the executable *sine.*

The 3rd line contains information about what is needed to make the object file *sinemain.o*. It needs the files *sinemain.cpp* and *sine.h*.
The 4th line is the action that will produce the *sinemain.o* object file. The source file *sinemain.cpp* needs to be compiled in order to produce the respective object file.

The 5th line contains information about what is needed to produce the object file *sine.o*. It needs the files s*ine.cpp* and *sine.h*.
The 6th line contains information about the action that will produce the *sine.o* object file. The source file *sine.cpp* needs to be compiled in order to produce the respective object file.

10. The *sine.h* file is a header file. It contains a function declaration/prototype. This is important for compiling because it tells the compiler what the function takes in (arguments), and what the function returns. The *sine.h* header file is in included/imported in *sinemain.cpp*. Essentially, it declares the sine function, so the compiler knows that the source file *sinemain.cpp* can use that function. The *sine.h* file is a requirement for *sine.o* because *sine.cpp* contains the definition (i.e. How the function works) for the sine function. Basically, this helps the compiler link the files. It's able to figure out that *sine.h* is a function prototype, and the definition for it is in *sine.cpp*.

The command *touch sine.h* changes the edit time of the file. In turn, this causes the makefile to run all commands over again, and everything needs to be re-compiled and linked, all object files and executables. This indicates that *sine.h* is compiled/linked with the other two files.