2G03 Homework 5, 2020

Due: Tuesday, Oct 27th (suggested), (Hard deadline: Thursday, Oct 29th)
The files are available on phys-ugrad in `/home/2G03/HW5`
Please follow the guidelines on code and homeworks. You will use a directory called *HW5* in your area for these programs and Makefiles for them. The grader will look there to locate and run your programs.

Some code and Makefiles are already provided. You can copy them all to your area (which also creates the directories) with:
`cp -r /home/2G03/HW5 ~/`
This directory already contains code. You should start with the C++ files there and change them rather than creating new files yourself.

For full credit you must provide C++ code for the exercise marked with `PROGRAM` and responses to ALL the questions below. A working copy of all code should present of `phys-ugrad` and all testing and running should be done there. You do not need to print out the Makefile. Responses to questions can be hand-written (e.g. scanned PDF) but a text file is preferred.

Numbers and Functions

Exercise `PROGRAM` *Functions*

In this exercise you will write a program to estimate the sine of a real number. The program is very similar to the line and gcd programs that were explored in class time. In particular, it is very like the function version of the gcd program.

I have provided an example main program called sinestandard.cpp that is listed below and a Makefile. This version uses the uses the standard library sine function: `sin`. You can compile it with `make sinestandard`.

file: sinestandard.cpp

```
#include <iostream>
#include <cmath>

int main()
{
// Program for estimating the sine of a user specified real number
// Uses standard sin function from cmath
// Filename sinestandard.cpp Src: JWW 2020

  float val;

  std::cout << "Welcome to the sine estimator program\n";
  std::cout << "Input a real number\n";
  std::cin >> val;

  std::cout << "Sine( " << val << " ) = " << sin(val) << "\n";
}
```

1) There is an extra include statement, `#include <cmath>` in the file sinestandard.cpp. Describe the purpose of this include statement. You can look up any outside references you like.

file: sine1file.cpp

```
#include <iostream>

float sine(float x)
{
// Function to estimate the sine of a real value x
// Your code goes here ...

}

int main()
{
// Program for estimating the sine of a user specified real number
// Filename sine1file.cpp Src: JWW 2020

  float val;

  std::cout << "Welcome to the sine estimator program\n";
  std::cout << "Input a real number\n";
  std::cin >> val;

  std::cout << "Sine( " << val << " ) = " << sine(val) << "\n";
}
```

2) I have provided a source file, sine1file.cpp in the directory you copied `/home/2G03/HW5` (see the box for a listing). This file contains the definition of a function `sine()` for which you need to write the code yourself. This definition states that this function returns a float value and expects a single float argument. In this sense it is a drop-in replacement for the standard `sin` function. All the function needs to do is to calculate the estimate for sine and return that value. Add code to do so using this Taylor series expansion for sine about x=0. Note that there is no C++ operator for power of. Just use multiplies for now.

$$\text{sine}(x) = x - \frac{1}{6}x^3 + \frac{1}{120}x^5. \tag{1}$$

3) In this case both the sine function and the main function are in a single file. This makes things easy for the compiler as long as the sine function appears before the main function in the file it is aware of it. You can compile this code using `make sine1file`. The Makefile includes the math library with `-lm` when linking to make sure regular `sin` works. Compile your program. It should have no error messages. It will create a runable program called `sine1file`.

4) Run your `sine1file` program to estimate the sine of 0.5, 1, 2 and 4 and report the values. Compare these estimates to a more precise estimate of the sine, such as using the sinestandard program.

5) Comment on the usefulness of a Taylor series like this once you have values not that close to 0 (such as 4).

6) Propose a smarter way to get estimates for values of x bigger than 1.5 or so.

7) It is awkward to paste all your code into a single file for every project. If you write functions you may use them for many projects. Therefore the standard approach is to put functions into separate files. The next exercise is to separate the code into two separate C++ files: `sine.cpp`, containing the sine function and `sinemain.cpp` containing the main program. There are templates for these files already provided in the directory.

A key difference is how the compiler is going to find information about the function sine. For this purpose I have provided a header file, sine.h. This header contains a prototype of the sine function you wrote yourself. Note how it contains the basic information from the function definition – what the arguments are and what it returns.

The Makefile contains rules to make this program as well. You can just type `make sine`. The runable program it creates is called `sine`.

8) Confirm that your program compiles without errors and gives the same results as `sine1file`.

```
sine: sinemain.o sine.o
c++ sinemain.o sine.o -o sine

sinemain.o: sinemain.cpp sine.h
c++ sinemain.cpp -c

sine.o: sine.cpp sine.h
c++ sine.cpp -c
```

9) The above excerpt is from the Makefile for the sine program. To compile your program from scratch it needs to use all these lines. Describe the purpose of the 6 lines in the above excerpt.

10) Why do you think sine.h appears in the Makefile like this? You can use the `touch sine.h` command to trick the make program into thinking sine.h has changed and see what it does differently.