HW #13: Multiple Files

Due dates: Sunday Apr 17th, 11:59 pm through Handin (https://secure.cse.msu.edu/handin)

Lab Assignment

Getting started

Change into the cse220 directory. Create a new directory called lab13.
Change into the new directory. Implement the program below in your lab13 directory.

Splitting into multiple files

You are to create a program that reads the dimensions of geometric shapes from command line and computes and outputs the area of the shape. Your program will consist of multiple files as follows.

Create a header file geometry.h and add to it the following:

- A marcro definition for the constant PI
- A prototype definition for the function computeCircleArea. This function takes as input the radius of a circle and returns its area
- A prototype definition for the function computeRectancleArea. This function takes as input the length and width of a rectangle and outputs its area.

Create a source file geometry.c and add to it the definition of the two functions.

Create your main program areas.c so it reads the parameters from command line. If the shape is a circle, the command line parameter should be the letter C followed by the radius. If the shape is a rectangle, the command line parameters should be the letter R followed by the length and the width. The following is an example of valid parameters.

```
>a.out R 12.5 14.2
>a.out C 3.6
>a.out C 10.2
```

Your program should compute the area according to the parameters given. If the program is given invalid parameters, it should display an error message to the user and exit. The outputs corresponding to the previous examples are:

```
R 177.5
C 40.7
C 326.8
```

Note that the numeric values read from command line are read as strings. Use the function `atof` defined as: *`double atof(const char *str)`* in stdlib.h to convert from string to float as needed.

Use the following to compile and build your program:

```
      gcc –o areas areas.c geometry.h geometry.c
```

Test your program on the given examples as well as some other invalid inputs.

Using a makefile

Create a makefile to build your program:

Add the following rule:
areas: areas.o geometry.o

```
      gcc -o areas areas.o geometry.o
```

Make sure the second line of your rule starts with tab and not spaces. Build your program by executing the makefile:

```
      make
```

Does your program build successfully? The building process needs to know about the dependencies.

Add the following rule to your makefile:

```
areas.o: areas.c geometry.h
      gcc –c areas.c
```

Try to build your program. Is it successful? Add the last missing rule to be able to build your program successfully.
Build your program again. You should get a message similar to:

```
      make: areas is up to date
```

Try the following command to simulate modifying areas.o. Remember, the touch command sets the timestamp of the file to the current time.

```
      touch areas.o
```

Now build your program again. What files are regenerated?
Now touch geometry.c, then build your program. What files are regenerated?
Now touch geometry.h, then build your program. What files are regenerated?

Submit through the handin system the following files: area.c, geometry.h, geometry.c, makefile

Handin

The "handin" system has options to allow you to review your files online and to download them. You Should always verify that you submitted the correct files and they were received by the handin system. You can submit files as many times as you like for a particular assignment. Handin will only keep the last version of each file. Remember to submit your files prior to the deadline as you won't be able to use handin if the deadline has passed.