**PROBLEM**:

1. Write a function that finds both volume and surface area of a rectangular box.
2. Write a makefile to pull all these files together.

Your function must use **pointer notation** (the address operator “&” and the indirection operator “\*”). This function will be in a *file* *separate* and on its own.

**The function prototype is**:

/\* Function to compute the volume and the surface area of a rectangular box \*/

**void find\_box\_values (FILE \*data\_out, FILE \*data\_in,**

**double l, double h, double w,**

**double \*vol, double \*s\_area);**

You will need the file **lab5.c** as your main/driver program for the function. This main program will call all the subfunctions:

* open\_out\_file.c
* open\_in\_file.c
* print\_headers.c
* find\_box\_values.c

**THE FORMULAS** (in ***algebraic*** notation)(must be translated to C notation):

(1) the volume of a rectangular box.

v = l \* h \* w

(2) the surface area of a rectangular box.

s\_area = 2(l\*w + h\*w + h\*l)

**TO GET THE FILES YOU NEED:**

First move to your class folder by typing: **cd csc60**

The following command will create a directory named **lab5** and put all the needed files into it below your csc60 directory.

Type: **cp -R /gaia/home/faculty/bielr/files\_csc60/lab5 .**

Spaces needed: (1) After the **cp *↑*** *Don’t miss the space & dot.*

(2) After the **-R**

(3) After the directory name at the end & before the dot.

After the files are in your account and you are still in **csc60**, you need to type: **chmod 755 lab5**

This will give permissions to the directory.

Next move into lab5 directory (**cd lab5**), and type: **chmod 644 \***

This will give permissions to the files.

Your new lab5 directory should now contain: lab5.c, lab5.dat, lab5.h, open\_in\_file.c, open\_out\_file.c, print\_headers.c,

**INPUT/OUTPUT DESCRIPTION**:

The **input** is a list of varying length of type double values in the file **lab5.dat**. Each record (or line) will have three values for the three measures of the box. The data file will have four sets of values.

The **output** is a chart showing the three measures of the box, the volume, and surface area.

**DEFINED OUTPUT APPEARANCE**:

Print statements included in the code, and require no changes, except for your name.

Your Name, Lab5.

Length Width Height Volume SurfaceArea

------ ----- ------ ------ -----------

3.70 5.00 4.20 77.70 110.08

6.80 3.00 5.90 120.36 156.44

ONLY the two lines are shown here. You should validate the correctness of the other lines.

**ALGORITHM DEVELOPMENT - Pseudo code**:

/\*-------------------------------------------------------------------------\*/

main /\* main is given to you as ***lab5.c***  \*/

Call function **open\_out\_file.** Will open lab5.txt, mentioned in lab5.h.

Call function **open\_in\_file** Will open lab5.dat, mentioned in lab5.h

Call function **print\_headers** Will print the column headers

Call function **find\_box\_values** You write this one.

Close the files.

/\*-------------------------------------------------------------------------\*/

/\* This code will reside in a file **find\_box\_values.c** \*/

/\* In a *separate file from the other functions* \*/

// You need to do a #include of **lab5.h**

void find\_box\_values(FILE \*data\_out, FILE \*data\_in, //fixed

double l, double h, double w,

double\*vol, double\*s\_area)

while(( do an fscanf to read the l, w, h) ==3)

| Calculate the **\*vol**

| Calculate the **\*s\_area**

|\_ Print the length, height, width, volume, surfaceArea

print one more empty line.

return

/\*-------------------------------------------------------------------------\*/

**REMINDERS**:

* You should look at all the provided files and get an idea of what code each file contains.
* Remember to put your name and Lab 5 in the comment header of your function, in your makefile, and in the output.
* Remember the **only** operator for multiplication is the asterisk (\*).
* You should examine the data file and confirm the correctness of the answer produced by your program.

**CREATING A MAKE FILE:**  Use the slides 13-14 of 5-UNIX as a reference. Also pasted at the end of this file.

lab5

lab5.o open\_in\_file.o open\_out\_file.o print\_headers.o find\_box\_values.o

lab5.c lab5.h open\_in\_file.c open\_out\_file.c print\_headers.c find\_box\_values.c

lab5.h lab5.h lab5.h lab5.h

* Type: **vim makefile** to create a makefile
* On the first lines, use “#” at the start of each line for comments of your name and lab5
* Write the first and final rule to link it all together.
* Line 1 of the rule: Put the name of the executable **lab5**, followed by a colon, followed by all the function names ending with a “.o”
* Line 2 of the rule: press: **tab**, then type: **gcc**. Enter the names of all the functions again ending with “.o”. Add in **-o lab5** for the executable name.
* Example from another program:

*radii: lab5.o find\_two\_radii.o*

*gcc lab5.o find\_two\_radii.o -o radii -lm*

* Next, we must figure out what to do if any of those files listed above need to be recompiled. The make utility will check the date of the **.c** file against the date of **.o** file. If they are out of sync, then the **.c** file will get recompiled. The next step is to create multiple rules to take care of each file. So, to do that……
  + Line 1 of the rule: put the name of the .o file followed by a colon. Then add the name of the **.c** and **.h** files that the **.o** file is dependent on.
  + Line 2 of the rule: type **gcc -c** then the name of the **.c** file
  + Example from another program:

*find\_two\_radii.o: find\_two\_radii.c lab5.h*

*gcc -c find\_two\_radii.c -lm*

**🡪 more on next page**

* We need to repeat the above so there is a rule for each file. An empty line between each rule makes for readability. A final example for this other program would be:

*#Your Name Lab 5*

*radii: lab5.o find\_two\_radii.o*

*gcc lab5.o find\_two\_radii.o -o radii -lm*

*lab5.o: lab5.c lab5.h*

*gcc -c lab5.c -lm*

*find\_two\_radii.o: find\_two\_radii.c lab5.h*

*gcc -c find\_two\_radii.c -lm*

**PREPARE YOUR FILE FOR GRADING:**

When all is well and correct,

Type: **script StudentName\_lab5.txt** [Script will keep a log of your session.]

Type: **touch lab5.h** to force a recompilation

Type: **make** to compile and link the code

Type: **lab5** to run the program to show the output of the program

(or whatever name you used for the executable)

Type: **cat lab5.txt** to see the output of your program

Type: **exit** to leave the script session

**Turn in your completed session:**

Go to Canvas and turn in:

1. makefile
2. find\_box\_values.c
3. lab5.h
4. print\_headers.c
5. your script session (StudentName\_lab5.txt).

**🡪 more on next page**

**Helpful slides:**

**Slide 13:**

*/\* Second pass at a makefile: \*/*

*/\* Look at its contents. We have no* ***p2.h*** *but it is included in light italics*

*to show where it would be placed. \*/*

*>****cat makefile***

# Your Name

power2: power2.o compute.o p2.h

gcc power2.o compute.o -o power2 -lm

power2.o: power2.c p2.h

gcc -c power2.c

compute.o: compute.c p2.h

gcc -c compute.c

/\* Run make using our new makefile \*/

[bielr@athena ~/csc60]68> **make**

gcc -c power2.c

gcc -c compute.c

gcc power2.o compute.o -o power2 -lm

[bielr@athena ~/csc60]69>

**Slide 14:**

/\* Helpful Comments \*/

* Start by opening **vim** and typing in the commands to a file named **makefile**. Close vim and then at the prompt, type: **make**
* When you enter vim, type: **:set list** This will show the non-printable characters:

^I = tab (That is a capital I.)

$ = end of line

* To reverse the setting, type: **:set list!**
* To create a tab on athena, you may have to hit the tab key twice in a row.