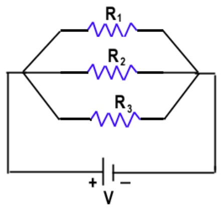
Basic Theory: When you started studying electronics, one of the first things you learn is how to deal with simple circuits that have a voltage supply, and resistors either in sequence or in parallel. Resistors in sequence are simple - their resistance just adds up, but resistors in parallel are slightly more complicated.

Consider the circuit on the right. In this circuit, there is a basic voltage source, and three resistors in parallel. We can calculate the resistance of the entire circuit using Ohm's Law, which in this case, can be stated as follows:



$$R_{Tot} = \frac{1}{\frac{1}{R_1} + \frac{1}{R_2} + \frac{1}{R_3}}$$

Suppose you are faced with lots of circuits that all have three parallel resistors, and you are asked to find the total resistance for all these circuits. It might be helpful to write a computer program that perform these calculations for you. In this lab, we will write such a program.

Follow these instructions to complete lab 02.

1. Log on to the Windows machine in front of you using your PODS userid and password, and follow the class web page instructions for “Connecting to a Virtual Desktop”. Open a Terminal Window.
2. Check to make sure your U-Drive is mounted with the command:
   mount | grep *userid*
   where *userid* is your PODS userid. If your U-Drive is working, you should see:
   “//bushare.bu.binghamton.edu/*userid*$”
   If you do not find such an entry, contact the CA. (You don’t have a network U-Drive.)
3. If you don’t already have a cs211 subdirectory of your home directory, make one with “mkdir cds211”, then change to that directory with “cd cs211”. Then make a lab02 subdirectory with “mkdir lab02”, and cd to that directory with “cd lab02”.
4. Create and edit a file in the lab02 subdirectory called “paraRes.c”. (Please use this exact file name to make the TA's job easier.) You can use the command “gedit paraRes.c&”. In this file, add the standard main function definition:

```
#include <stdio.h>

int main() {
        // Code Goes Here
        return 0;
}
```

5. Replace the “// Code goes here” comment with a program to calculate parallel resistance. In order to do so:
   a. you first need to get the values for R_1, R_2, and R_3. These values can be kept in three variables - call them whatever you want, but I called mine “r1”,”r2”, and “r3”. Resistance is normally specified in Ohms, and it is typical to keep track of Ohms using a value with a decimal point, so we should make these variables of type “float”.

   b. To get a float value from the user of your program, first prompt the user, telling her what you are asking for using the built-in function "printf", and then get the value from the keyboard using the built-in function scanf. For instance, to get one value, you might use the following code:

```
printf("Enter the value of the first resistor (in Ohms) :> ");
scanf(" %f",&r1);
```

   c. Once you have collected values for all three resistors, calculate and print the total resistance using Ohm's law. (A little algebra is required to find R_{Tot}, but you can figure that part out.) The division operator is "/", the addition operator is "+", and unless there are parenthesis, C will always perform division and multiplication before performing addition or subtraction.

   d. Print out your final value for R_{Tot} using a printf statement that contains a %f.

6. Save your results, and compile with the command

```
gcc -g -Wall -o paraRes paraRes.c
```

   If you get any messages, fix the problems indicated by the message by changing the C program, saving the changes, and re-running the gcc command.

7. Once your program compiles with no error messages, test your command by typing:

```
./paraRes
```

   Here is an example from a correct paraRes program:

```
~/cs211/lab02>gcc -g -Wall -o paraRes paraRes.c
~/cs211/lab02>./paraRes
Enter the resistance of the first resistor (in Ohms) :> 1.0
Enter the resistance of the second resistor (in Ohms) :> 1.0
Enter the resistance of the third resistor (in Ohms) :> 1.0
The total resistance is 0.333333 Ohms
~/cs211/lab02>
```

   Try several different test cases to make sure you program works correctly.
   Hint: Both 0, meaning no resistance and "inf" for infinity, meaning total resistance (no current) are considered valid input values for one or more of the resistors. Does your program print out the correct results using these values as well as normal numbers?

8. Open a Web Browser in the Linux window, and navigate to myCourses/CS-211 A 0. Click on "Content/Lab Submissions/Lab 02 Submission". On the submission page, attach "paraRes.c". Don't forget to click on the "Submit" button at the bottom left of the page.

9. Close all the applications running in your Linux Window, exit Linux by hitting the On/Off button, shut down the web browser in Windows, and sign off from Windows. You're done!

Grading Criteria:

- 10 points (full credit) if everything is correct and your program compiles and runs correctly
- -1 point if there are compiler warning messages with the -Wall gcc option
- -2 points if the submitted file is not "paraRes.c"
- -5 points if there are compiler error messages
- -2 points for each value where paraRes does not print the correct total resistance. There will be three test cases with valid input values.
- -2 points per 24 hours if the submission is late