NCSU ECE 209 Sections 602
Exam 2 Fall 2009
12 November, 2009

This is a closed book and closed notes exam. It is to be turned in by 5:45 pm. Calculators, PDA's, cell phones, and any other electronic or communication devices may not be used during this exam.

Please read and sign the following statement:

I have neither given not received unauthorized assistance on this test.

Name: _____

If you want partial credit for imperfect answers, explain the reason for your answer!

Problem 1 (12 points) Pointers
For the following block of C code, draw the pointers as they exist when the block is executed and write the output that will be printed when the block is executed. This would be a good place to explain your work.

```
{
     int i, j ;
     int m[3] ;
     int *p, *q ;
     i = 1 ;
     j = 2 ;
     m[0] = 3 ;
     m[1] = 4 ;
     m[2] = 5 ;
     p = &m[1] ;
     q = &i ;
     *p = *q ;
     printf("%d %d %d\n",
             m[0], m[1], m[2]) ;
     q = p ;
     ++p ;
     *q = j ;
     printf("%d %d %d\n",
             m[0], m[1], m[2]) ;
     ++*q ;
     m[0] = *p ;
     printf("%d %d %d\n",
             m[0], m[1], m[2]) ;
}
```

Problem 2 (12 points)
When two resistors with resistance R_1 and R_2 are placed in series, the combined resistance is $R_1 + R_2$. When these same two resistors are placed in parallel, the combined resistance is $1 / (1/R_1 + 1/R_2)$.

This time you are completely on your own. You must implement two C functions, each with two `double` arguments. One should compute the series resistance of two resistors; and the other, the parallel resistance.

First, write C prototypes for your two functions. (That should be one line each.)

Finally, write the actual functions themselves. (Each only needs a single `return` statement.)

Problem 3 (16 points)
Now write a program that prompts for two resistor values and then computes the four resistances and prints them in the following order (1) parallel resistance, (2) smaller input resistance, (3) large input resistance, and (4) series resistance. Here's an example of how your program might interact with a "user". The user's input is underlined.

```
Enter two resistor values in ohms:
200 100
The possible resistances are
      67 ohms
     100 ohms
     200 ohms
     300 ohms
```

Try to make your output neat, like that shown above. Although you are storing ohmage values as `double`s, write ohmage as whole numbers. After all, even the best tolerance is 1%. In your solution, you do not need to verify input. If the user types "`440k`" for the resistor value, your program is allowed to crash and burn. You also don't need to check if the user enters the same resistance value twice. However, your program should sort the two resistances, so that it can print the smaller before the larger.

Write your program on the following page.

```
/* Problem 3 solution */

#include <stdio.h>
#include <stdlib.h>

/* Stick your two prototypes right here */



/* Do not put function definitions on this page! */

int main(int argc, char **argv) {









    return (EXIT_SUCCESS) ;
}
```

There's no reason to limit ourselves to just two resistors!

Problem 4 (12 points)
Suppose you have a collection of 10 resistors. You could store them all in an array. Write a C declaration for a `double` array suitable for holding the ohmage of 10 resistors.

Next, write a declaration and initialization for a `double` array storing the ohmage for five resistors rated at 109, 209, 1060, 1776, and 2009.

Problem 5 (12 points)
Now suppose you are given a package of 1000 resistors such that the ohmage of the *i*'th resistor of the package is *i* 500Ω. (That is, the values are 500Ω, 1000Ω, 1500Ω, …, 500000Ω.)

Write a declaration for this array along with a C loop to initialize it to the values described above.

Problem 6 (12 points)
It's possible to compute the series resistance of a collection of resistors simply by summing them up (as in ΣR_i). Write a function that adds up the values stored in an array and returns the sum as the series resistance. This function will be passed both the array and its size.

```
double Series(double *R, int N) {




}
```

Problem 7 (12 points)
The parallel resistance of a collection of resistors is a tad harder. It's the reciprocal of the sum of the reciprocals of the resistors, *i.e.*, $1 / \Sigma (1/R_i)$. Now write a function for the parallel resistance. In C, The parallel solution function is not much longer than the series solution function. In fact, it can be only four characters longer.

```
double Parallel(double *R, int N) {




}
```

Problem 8 (12 points)
Each of the following two `for` loops print numbers. For each loop write in the five boxes below the loop the first four lines printed by loop. If the loop prints less than four lines, fill in a box for each line that is printed. Assume that `i` has already been declared as an `int` variables before each loop. **Explain your work for partial credit!**

```
for (i=1; i<= 10; i=i*i+1) {
  printf("%d\n", i) ;
```


```
for (i=1; i<10; i=i+3)
  if (i%2 && i%3==1)
    printf("%d\n", i) ;
```
