EE 285 Lab 8

A. The resistor program with functions (50 points)

Re-write the earlier resistor program, but use functions. Function prototypes and the main function are listed below. Your job is to write the various functions. The program assumes that there will always be three resistors. Basically, the program should: a) ask if the calculation will be series or parallel, b) ask the user to enter the three resistor values, c) perform the correct calculation, and d) print the result. You should not alter the main part of the program. If needed, you can make minor modifications to the prototype definitions if that helps you write your functions more efficiently.

Comments: Yes, it would be much better to use arrays. The code would be cleaner, and there could be any number of resistors allowed. But we are not quite ready for that yet. Most of the functions needed here will be almost trivial — perhaps only one line. The point is to begin viewing programs as a collection of basic functions. We will soon get to the point where the functions (and hence the overall programs) are more complicated.

```
//EE 285 - lab 8A

#include <stdio.h>

char seriesOrParallel( void );
double enterRValue( void );
double seriesCalculation( double, double, double );
double parallelCalculation( double, double, double );
void printResult( char, double );

int main(){
   double resistor1, resistor2, resistor3, result = 0;
   char a;

   a = seriesOrParallel();

   resistor1 = enterRValue();
   resistor2 = enterRValue();
   resistor3 = enterRValue();

   if( a == 's' ){
      result = seriesCalculation( resistor1, resistor2, resistor3 );
   }
   else if (a == 'p' ){
     result = parallelCalculation( resistor1, resistor2, resistor3 );
   }

   printResult( a, result );

   return 0;
}
```

B. A prime number checking function (25 points)

Create a function that will take a positive integer *x* and check to determine whether it is a prime number. A simple (but perhaps not terribly efficient) approach is to divide the number by every integer between 2 and *x*/2. If any division has a remainder of 0, it is not a prime number. The prototype and main function are given below — you must write the function. Your function should check for "primeness" and then print a statement saying whether the number is prime or not.

```
//EE 285 - lab 8B

#include <stdio.h>

void checkPrimeNumber( int );

int main(){

   int x;
   printf( "Enter the number to check: " );

   scanf( "%d", &x );

   checkPrimeNumber( x );

   return 0;
}
```

C. Quiz (25 points)

As usual, there will be a short quiz. The quiz will be on functions. (No surprise there.)