CS16, UCSB
Pre-lab #4: Worth 50% of Lab 4 score (50 total points)

Print this form, staple loose pages together, and write your answers on it.

Accepted: On paper, in lab Wednesday, July 25.

Name (2 pts): _____

Email (2 pts): _____

Lab section (2 pts) Circle one: 2:00 3:30

If you have the book, read the rest of Chapter 3, especially sections 3.4 (Loop Structures) and 3.6 (Data Files). You may skim through sections 3.7 and 3.8, but do learn about summation notation (p. 129) and carefully examine the program developed on pp. 133-136. Then answer the following items.

1. (24 pts; 8 pts each) Solve the following problem three different ways, by first applying a while loop, then a do/while loop, and finally a for loop. The problem is to find the sum of all integers from m through n. For example, if m is 4 and n is 7, this sum is 4+5+6+7 = 22. You may assume that m will always be less than or equal to n. There is no need to print anything. Just find the sum. Imagine the following statements have already been executed before your solutions start:

```
int m, n; /* start and end values */
int i; /* a variable you can use to develop your
        * solutions */
int sum = 0; /* store the result in this variable;
              * already initialized */

printf("enter m, n: ");
scanf("%i %i", &m, &n); /* assume that m <= n */
```

a. Write your while loop solution here.

b. Write your do/while loop solution here.

c. Write your for loop solution here.

2. (5 pts) You just found the sum of all integers from `m` through `n` three ways. Such a sum can be neatly described by "summation notation" - describe it that way. A potential useful reference is here: http://en.wikipedia.org/wiki/Summation#Capital-sigma_notation

3. (15 pts) As a reminder, strings in C are represented in memory as a series of characters (represented by the single byte `char` type), terminated by a special null byte (‘\0’). Consider the following code that creates a string “foobar” and makes it accessible through the variable `string`:

```
char* string = “foobar”;
```

In C, individual characters of a string can be accessed using a special square brackets notation. For example, using the same variable `string` from above:

```
string[ 0 ] // returns the first character of the string
            // this returns 'f' for the string "foobar"
string[ 1 ] // returns the second character ('o')
...
string[ 6 ] // returns the null byte ('\0') that
            // terminates the string
```

Let `s` be a variable of type `char*` that allows access to some arbitrary string. This variable `s` works in the exact same manner as the variable `string` above, except now the string's contents are unknown. (It could be "foo", "bar", "foobar", "house", "plane", etc.)

Consider the following code:

```
char* s = ...; // don't know what ... is
int length = 0;
```

Write code below that will ultimately set the variable `length` to the number of characters long the string is, not counting the terminating null byte. For example, for the string "foo", `length` should equal 3 by the end of the code, and for the string "foobar" `length` should equal 6 by the end of the code. (Hint: A loop is needed for part of the solution, and a `while` loop is probably the most natural choice.)

Pre-lab End. Adapted from Michael Costanzo by Kyle Dewey.