

## CISC181H Spring 2009 Lab02

- Write a program for each of the following problems. Be sure to save every separate program. All programs must be properly commented and indented (see Assignment Standards on the class website). Ask your TA for guidance.
- Think about what you want to do, and then look up how to do it in Savitch. For some problems I included the C math library in my code so I could use math functions from there, esp `ceil` and `floor`.
- Name each program `lab02.n.cc`, where `n` is the number in the list below. For example, the name of the file for the first will be `lab02.1.cc`. Put the files in your `lab02` directory (see Anderson for instructions on how to make a directory).
- Do not be afraid to write multiple functions for a single problem. Short, simple functions are much more likely to work, and tend to be easier to write.
- Design on paper *before* you start coding. This is the time to break tasks into multiple functions.
- C++ has a type called “string”. For a lot of good reasons, we aren’t using it yet; we are using the C-strings we discussed in class.

## Programs

1. Solve the diagonal line of asterisks problem from lab01 recursively. You might use the case where *height* = 1 as a base case, or *height* = 0. Think ahead about what the parameters need to be (i.e. what is “state” in this problem?)
2. Change the order of the statements in program 1 so that the diagonal line now prints in the other direction. I think you will only have to move one statement to get this to work.
3. Write a function (no parameters) that reads two numbers entered by a user and returns the sum. All vars inside the function should be dynamically allocated. Call your function three times from main.
4. The C-string functions all rely on the presence of the null terminating character in a char array. Typically they work using *sentinel loops*<sup>1</sup>  
Demonstrate the use of the C-style string functions `strlen`, `strcmp`, `strcpy`, `strcat`, and `strtok` from the `cstring` library. Display “before and after” data if appropriate. The GNU C library online is a useful source for this, as is Chapter 9 of your text.  
If you wish, write your own versions of these functions for practice. At least try drawing them on paper; these functions make good exam questions to test knowledge of arrays and sentinel loops.
5. Write the C-string function `strtok`. Use char functions from `ctype.h` to make your life easier (see p. 919 in your text). This function traverses a char array and uses all the concepts you need to work with C strings. Assume that tokens contain letters or numbers, and that the function only needs to look for delimiters specified in the second string. Be sure you are comfortable using this function before you try to write it.

---

<sup>1</sup>A sentinel is a value that will stop the loop; be careful that it is not part of the set of valid data to be entered!

You should have a total of 5 programs named lab02.1.cc to lab02.5.cc. Make a single script file (see lab00 for the instructions) where you cat, compile, and run each one in its final form.

Submit all 5 program files *and* your script on WebCT by midnight before your next lab. Give the paper version of the complete script file **only** on paper to your TA at the **beginning** of your next lab. Note: Cat, compile, and run each program in order - do *not* cat all programs, then compile, etc.