Lab 4

Each lab will begin with a recap of last lab and a brief demonstration by the TAs for the core concepts examined in this lab. As such, this document will not serve to tell you everything the TAs will in the demo. It is highly encouraged that you ask questions and take notes.

In order to get credit for the lab, you need to be checked off by the end of lab. For non-zero labs, you can earn a maximum of 3 points for lab work completed outside of lab time, but you must finish the lab before the next lab. For extenuating circumstances, contact your lab TAs and the course instructor.

Pair Programming: In this lab, you can choose a partner for pair programming. **You must be checked off together. You only need one computer for pair programming.** One person will be the driver, who controls the computer and codes, while the other is the navigator, who observes and advises. After 20 minutes, you will switch driver and navigator, continuing this pattern until the task is complete. Please read more about pair programming and the benefits: Pair Programming Handout

The Linux configuration portion of this lab should be completed independently, but the remainder of the lab can be done using pair programming.

Demo

Follow along with your TAs as they show you how to construct a function for determining if a string contains an 'a' character. This should be very similar to the demo code that was written in class to detect whether a binary string contained only 1's and 0's.

```
bool checkforA(string test)
Input: a string
Output: return true if it has an 'a', false otherwise
```

By the end of demo you should be able to identify (6) things: a parameter, an argument, a return type, a function call, a function declaration, and a function definition.

(1 pt) VIM or Nano Configuration

Some students have come to office hours or asked the TAs for advice on configuring Vim or Nano. If you have not already configured your editor of choice this exercise will help explain the procedure.

Configuration files are available that will enable features such as syntax highlighting and easier-to-read indentation. This **configuration needs to be repeated by each student** in the pair. For this part of the lab, you will have two options.

If you prefer Vim, you may run the following command to download a vimrc file:

```
wget http://web.engr.oregonstate.edu/~hilbertt/VIMRC/vimrc.txt -O ~/.vimrc
```

You can edit this file by executing:

```
vim ~/.vimrc
```

This configuration file covers the basics. There are many more commands you can insert in this file. Here is a reference guide to some of these additional features: http://vimdoc.sourceforge.net/htmldoc/starting.html

If you prefer Nano, you may run the following command to configure syntax highlighting:
`wget http://web.engr.oregonstate.edu/~goinsj/resources/CS161/.nanorc -O ~/.nanorc`

You can edit this file by executing:
`nano ~/.nanorc`

As you might expect, there are additional configuration settings available for Nano as well.
You can see these settings using the man page:
`man nanorc`

(2 pt) Design atoi
atoi() is a common function which takes a character and returns its decimal ASCII value. Start by designing how this function will work. It should take any character found on the ASCII chart (http://www.asciitable.com/) and return the decimal value.

```
/*********************************************************************
** Function: a_to_i
** Description: turns a character into a decimal value
** Parameters: char character
** Pre-Conditions: the input is a character
** Post-Conditions: returns the decimal value of the character
*********************************************************************/
```

(2 pt) Implement atoi
Write the a_to_i() function based on your design. Test your function thoroughly.
Will your function properly return the decimal value of: 'A', '1', 'b', '/', ' ', etc.?

(2 pt) Design itoa
Similar to a_to_i(), i_to_a() takes an integer and returns the character associated with that value. Design this function.

```
/*********************************************************************
** Function: i_to_a
** Description: turns a decimal value into a character value
** Parameters: int decimal
** Pre-Conditions: the input is an integer
** Post-Conditions: returns the character which represents the decimal value
*********************************************************************/
```

(2 pt) Implement itoa
Write the i_to_a() function. Test your function thoroughly.
Will it return the correct character value for 127, 65, 97, etc.?
For this exercise, you can assume that the input will not be less than 0 or greater than 127.

(1 pts) Testing and Debugging

Download the following cpp file:
http://web.engr.oregonstate.edu/~goinsj/resources/CS161/demos/buggycode.cpp

You must find as many bugs as possible and fix them. Some are logic errors, some are syntax errors. Hint: there are 17 bugs, some obvious, some more complex. **Make sure you re-compile and run your program after you make a single fix to a mistake to make sure you actually fixed the mistake, didn't introduction new errors, and/or eliminated other errors as a result of the fix.**