Week 4 Lab

Programming in C

This week's lab is a lot shorter than the previous weeks' labs. This is to allow you enough time to speak to your demonstrator about the assignment if you are not clear on the requirements.

Task 1. Functions and iteration

Write a C function `void reverse(char *string)` that takes as its parameter a null-terminated string and reverses it in place. That is, it should only require O(1) space aside from its input string. You may use any function you wish from the string library to help you – use `man string` Hint: when reversing the string, the terminating null character (`'\0'`) should remain where it is.

Once that function is complete, write a `main()` function that takes one command-line argument, calls `reverse()` to reverse it, and then outputs the reversed string to `stdout`.

Task 2. Dynamic data structures and `valgrind`

Download the following files from Moodle:

- `bst.h` - a header file for a binary search tree that uses C strings as keys
- `spellcheck.c` - the source code for a program that uses a binary search tree to spellcheck a text file, using functions declared in bst.h
- `simple-english.dict` - a dictionary containing some of the most common English words
- `simple-sample.txt` - a very simple sample to try as a test input
- `sample-text.txt` - a sample file to use as a test input

Create a new file, `bst.c`, that implements the functions declared in bst.h, and use it to compile the `spellcheck` executable. Run it using `simple-sample.txt` and `simple-english.dict` and confirm it works. Then try the `sample-text.txt` file, with both `simple-english.dict` and `/usr/share/dict/words` on your VM.

Once you have convinced yourself that your program works, install the `valgrind` package using your favourite package manager. Read the Valgrind Quick Start Guide at `http://valgrind.org/docs/manual/quick-start.html`. If you like, copy and compile the short buggy program at that page and verify that `valgrind` finds the errors in it. Run `spellcheck` under `valgrind` with the leak checking option on and see if your program leaks memory. Fix any leaks you find.

Hint: freeing a binary search tree won't be as simple as just freeing the pointer to its root node.