Programming Assignment #1

Files due Wednesday, January 20, 2016 at 11:59 pm

This assignment gives you some practice developing a simple program under Linux. The goal is to implement 6 functions that will create, manipulate, and destroy linked lists. The goal is not to gain any great insight into linked-lists, but the process of coding on an operating system that is new to most students. To keep the complexity to a minimum, the lists will be singly-linked. This is the definition of the node that will be used by the single-linked list:

```
#define LABEL_SIZE 16
typedef struct sll_node
{
  char label[LABEL_SIZE];
  int value;
  struct sll_node* next;
}sll_node;
```

Each node (*sll_node*) in a list contains a text label, an integer, and a pointer to another `sll_node`. This node structure is intentionally kept simple so you can focus on the list aspect of this assignment. This is the interface:

```
sll_node *sll_add(sll_node *list, int value, const char *label);
sll_node *sll_remove(sll_node *list, int search_value);
sll_node *sll_insert_before(sll_node *list, int search_value, int value, const char *label);
void sll_insert_after(sll_node *list, int search_value, int value, const char *label);
void sll_destroy(sll_node *list);
void sll_dump(const sll_node *list);
```

Since you will be programming in C, there are no classes, so all of the functions will take a list (a pointer to the first node) as their first parameter. Three of the functions will return the list that was passed in. At first glance, this may seem redundant but it is required as was explained in CS120. The reason is due to the fact that, if a function changes the first node (by an add, insert, or remove) then the function must return this new "first" node to the caller. We also saw that an alternative approach is for the functions to take a pointer to a pointer to a node instead of returning a node.

Function	Description
`sll_add`	Adds a new node to the end of the list. The first time this function is called, the list will be NULL. Return the head of the list.
`sll_remove`	Given a value, find the first occurrence of it in the list and remove it. Be sure to free the memory that was allocated for the node. If the list does not contain the value, nothing is removed. Return the head of the list.
`sll_insert_before`	Given a value to search for, find the first occurrence of it in the list and insert a new node before it. If the list does not contain the value, nothing is inserted. Return the head of the list.
`sll_insert_after`	Given a value to search for, find the first occurrence of it in the list and insert a new node after it. If the list does not contain the value, nothing is inserted.
`sll_destroy`	Walk the list and free all of the nodes.
`sll_dump`	Print out the list. **This is implemented for you. Do not change it.**

The assignment is relatively straight-forward and I don't expect any third-semester Digipen student to struggle with this. This is kind of a practice assignment (however, it will be graded) for you to gain more familiarity with developing under the Linux environment. You will use singly-linked lists in a future assignment, so successful completion of this assignment will help you with that assignment. That future assignment has additional complexities that you will need to focus your attention on, not the linked-list part. So, successfully completing this assignment will give you a head-start on the future assignment.

Other criteria

1. If you create other helper functions, they must be tagged as **`static`.** In C++, you would put them either in the private section of a class, or in an unnamed namespace in the implementation file. Neither of those exist in C, so you must use the **`static`** keyword.
2. Be sure to free all of the memory that you allocate. This should be easy for you to verify because you are required to use *valgrind* to look for memory leaks. I will post a command line that you should use.

More on back →

Deliverables
You must submit the implementation file (`slist.c`) and the `typescript` file. These files must be zipped up and submitted to the appropriate submission area. Notice that you are not submitting the header file, so don't make any changes to that file.

Files	Description
`slist.c`	The implementation file. All implementation for the functions goes here. You must document the file (file header comments) and all functions (function header comments) using the appropriate Doxygen tags.
`typescript`	This is the session that was captured with the `script` command.

Usual stuff
Your code must compile (using the compilers specified) to receive credit. The code must be formatted as per the documentation on the website.

Make sure your name and other info is on all documents.