CPSC 351, Operating Systems Concepts

Homework #3, Chapter 7 Synchronization Thread Safe Stack

1) (*20 pts*) The C program stack-ptr.c (provided) contains an implementation of a stack using a linked list. An example of its use is as follows:

```
StackNode *top = NULL;
...
push(5, &top);
push(10, &top);
push(15, &top);
...
int value = pop(&top);
value     = pop(&top);
value     = pop(&top);
...
```

This program currently has a race condition and is not appropriate for a concurrent environment. Using Pthreads mutex locks (section 7.3.1), fix the race conditions. Test your now-thread-safe stack by creating 200 concurrent threads in main() that intermix pushing and popping values.

- Use a loop in main() to create all those threads. Apply all the things you've learned about creating and joining threads from previous chapters.
- Write one testStack function, and use it as the entry point for each thread.
- The testStack function should intermix 3 push operations with 3 pop operations in a loop that executes 500 times.
- All threads use the same stack.
- `gcc -pthread stack-ptr.c -o stack-ptr` is an example command to compile and link your program
- If you're up for it, you may rewrite the provided C program into well-formed C++ code using proper Object Oriented Design concepts, but you must maintain the dynamically allocated linked list concept, and you must still use the Pthreads mutex locks.
- Submit your solution to TITANium.