Assignment 4: Dynamic List

Topics:	Doubly-linked lists, Dynamic Memory, Valgrind
Date assigned:	Friday, October 2, 2015
Date due:	Part 1: Monday, October 12, 2015, 11:59 pm Part 2: Friday, October 16, 2015, 11:59 pm
Points:	40

For this assignment, you are to implement the List ADT in a file called list.c using the header file list.h. You can find this header file on Zeus in **/home/CS300Public/2015**. All of the data structures and function prototypes are defined in list.h. Further, each function prototype has been described to the point that you should be able to implement each list function in the file list.c.

In addition to implementing the list data structure, you must provide a Makefile and test driver (**listdriver.c** that produces an executable named **listdriver**) that thoroughly tests your list functions. The listDriver should display to the screen a series of SUCCESS or FAILURE messages with enough description that a user can quickly spot broken list functionality.

You may add any helper functions you need to list.c. You may not alter list.h in anyway.

1. Your code is to be written in C using Eclipse. Programs written in other environments will not be graded. Create an Eclipse project named **DynamicList**. This project must contain the directories: src, include, and bin.
2. The Makefile must contain the necessary targets to build the listdriver as well as a clean, valgrind, and tarball targets. Typing **make** on the command line must build listdriver.
3. Your program must not have any Valgrind errors.
4. Submit a color, double-sided, stapled packet of code by that same deadline. The packet must be in the following order:

   List Driver (.h then .c if you have both, otherwise just .c)
   list.c (do not print list.h)
   Any extra .h/.c modules
   Makefile
5. Test one function at a time. This will lessen your level of frustration greatly.
6. You are to use the coding guidelines from V6.3 of the coding standards.

Goals for this assignment:

1. Code and test your program one function at a time.
2. Write efficient/clean code
3. Use the debugger and Valgrind to effectively develop a correct solution
4. Thoroughly test your code.

The list.h header file as well as a list of ERRORCODEs that each function can produce are on the Web page. Further, the ERRORCODEs are listed in order of precedence. If a function could produce multiple ERRORCODEs, the function must return the ERRORCODE highest on the list.

The meaning of the ERRORCODEs are given in the header file.

Since the interface for the list may be unclear, here is a very small example of how to walk a list and print out every element. For brevity, no error checking is done.

```
List sTheList;
int size;
bool bIsNotEmpty;
DATATYPE sTheData;
int i;

lstCreate (&sTheList);
sTheData.charValue = 'A';
lstInsertAfter (&sTheList, sTheData);
sTheData.charValue = 'B';
lstInsertAfter (&sTheList, sTheData);
lstSize (&sTheList, &size);
lstFirst (&sTheList, &sTheData);
for (i = 0; i < size; ++i)
{
  lstNext (&sTheList, &sTheData);
  printf ("%c", sTheData.charValue);
}
```

Part A: You must complete lstCreate, lstDispose, lstSize, lstIsFull, lstIsEmpty, lstFirst, lstInsertAfter, lstNext. Submit your assignment as cs300_4_A_PUNetID.tar.gz

Part B: Complete the entire assignment. Submit it as cs300_4_B_PUNetID.tar.gz.

I expect you to start this project early. My solution to this assignment required over 1000 non-commented lines of code.