

Software Systems

Mini Assignment #5

Due: November 26, 2017 on myCourses at 23:30

Linked Lists and Modular Programming

This assignment explores malloc, makefile, modular programming, and git.

The central problem you will solve is a linked-list like the one shown in the lecture slides. You do not need to use the lecture slides.

Construct a modular program from three source files: **main.c**, **list.c** and **list.h**. The file **main.c** will be used to call the functions provided by the file **list.c**. The file **list.c** builds a private linked-list within the **list.c** named space. The file **list.h** contains the definition of the list's node structure. Use **#ifndef** to protect the possible multiple inclusion of **list.h**, as we talked about in class.

Use a **makefile** to compile your assignment. Use git to version control the development of your assignment. To prove that you used a makefile **submit your makefile file**. To prove that you used git, **save** the **output** from **git log** into a text file and **submit** that file.

The program will run as follows:

- The **main()** function uses a while loop accepting only positive integer numbers ($n > 0$) from the user. The loop terminates once the user enters a non-positive value. The user is prompted for each number one at a time.
- In the loop, **main()** calls functions from the **list.o** named space that creates a private linked list using **malloc**.
- The following function signatures are supported by **list.c**:
 - **void newList();**
 - This function assumes there is a private global linked-list pointer in **list.c** called **head** that is used to point to the beginning of a linked-list. This function call simply initializes this pointer to **NULL**.
 - **int addNode(int value);**
 - This function **mallocs a new node and copies the parameter value into the node**. **This function then adds the node to the head of the linked-list**. The function ends by returning **true** for success and **false** for failure. A node has the following structure: `struct NODE { int data; struct NODE *next; }.`
 - **void prettyPrint();**
 - This function assumes a global pointer called **head** exists. Using this head pointer is traverses the linked-list printing all the values stored in the list.
- When the loop terminates the program prints to screen all the numbers in the list. The output should be in reverse order.
- The program then terminates.
- **Do NOT use recursion.**

check mycourses

WHAT TO HAND IN

Submit your .c files, makefile, and gitlog.txt files. **and list.h**
The TA will compile and run your program on the Trottier computers.

HOW IT WILL BE GRADED

Points removed for bad practices:

- -1 for not following instructions
- -1 for not indenting, spacing, and/or commenting
- -1 for not using good variable names

This assignment is worth 20 points:

- +2 main()
- +4 List.c functions as described
- +4 Private data structure
- +4 #ifndef and list.h usage
- +2 GIT usage
- +4 Correct makefile usage