CSE 109: Systems Programming

Fall 2017

Program 3: **Due on Monday, October 2nd at 9pm on CourseSite.**

Collaboration Reminder:

1. You must submit your own work.
2. In particular, you may not:
   (a) Show your code to any of your classmates
   (b) Look at or copy anyone else's code
   (c) Copy material found on the internet
   (d) Work together on an assignment

Assignment: Preparation

1. Make a *Prog3* directory in your class folder.
2. Create the files *List.c* and *List.h*
3. Copy the provided *prog3.c* file from ∼jloew/CSE109/

   Do not edit this file unless told to do so. Do not edit it.
4. All your source code files must have the comment block as shown at the end of this document. All files must be contained in your *Prog3* directory.

Assignment:
You will be creating both a header and an implementation file for this assignment.

1. *List.h*
   (a) Use conditional compilation to prevent List.h from being included more than once.
   (b) Declare the *Node* struct.

   Do not use a typedef, nor an anonymous struct.

(c) Include as few *#include*s as possible.

    List.c will contains more *#include*s since it is doing the real work.

(d) Declare the following function prototypes (You may give may to the arguments if desired.

    i. struct Node* constructList(int);
    ii. struct Node* destroyList(struct Node *);
    iii. size_t size(struct Node *);
    iv. struct Node* insert(struct Node *, int);
    v. int find(struct Node *, int);
    vi. int traverse(struct Node *, char *);
    vii. struct Node* removeItem(struct Node *, int);
    viii. char* traverse2(struct Node *);

2. *List.c*

    (a) Use conditional compilation to prevent List.c from being included more than once.

        It doesn't matter in this case but do it to get used to it.

    (b) Define the *Node* struct.it will contain an int and a *Node* pointer within it.

    (c) Define all the functions specified in *List.h*.

        i. struct Node* constructList(int): Creates a new *Node* object, fills it with the provided value and has it point to NULL. Return the newly allocated *Node* object.
        ii. struct Node* destroyList(struct Node *): Deallocates the provided *Node* object. If that object pointed to a *Node* object, deallocate it with destroyList and set the pointer to NULL. Return NULL.
        iii. size_t size(struct Node *): Returns the size of the *List* that starts at the given *Node*. Specifically, the number of reachable *Node*s from that given starting point.
        iv. struct Node* insert(struct Node *, int): Adds the given value to the end of the *List* that is specified by the given *Node* - which acts as the starting, or beginning of the *List*. If the given *List* is NULL, create a new *List* from this insertion. In either case, return a pointer to the *Node* that refers to the beginning of this *List*.

v. int find(struct Node *, int): If the *List* specified by the given *Node* contains the value, return 1. Otherwise, return 0.

vi. int traverse(struct Node *, char *): Using the given *List* specified by the *Node*, generate a string containing all the elements of the list in a comma-separated format. Put that string into the *char ** that is provided. Assume there is enough space in the *char **.

If the *List* is empty, the resulting string would be empty.

Otherwise, comma-separate the values using a comma follwed by a space. There must be no leading space or comma.

vii. struct Node* removeItem(struct Node *, int): Remove **all** occurrences of the given value from the *List*. Return a pointer to the first element of the remaining *List*.

viii. char* traverse2(struct Node *): Just like traverse but you are not given a *char **. You must allocate a sufficient *char ** and put the traversal into it. The user will be responsible for freeing the memory.

Do not blindly allocate memory for this. Allocate it based on how much is needed, or the potential maximum amount needed for that particular *List*.

Testing:

1. *prog3.c* references the expected code in *List.c* and *List.h*. Look through it to figure out how the command line arguments can be used to test your code.

2. You will need to use multiple steps to compile your code since you will have more than one .c source file:

```
module load gcc-7.1.0

gcc -Wall -g -c prog3.c

gcc -Wall -g -c List.c

gcc -Wall -g -o List prog3.o List.o
```

Submission:

1. Once ready to submit, you can package up the assignment as a .tgz file

```
tar -czvf Prog3.tgz Prog3
```

You must use this command in the directory that contains the *Prog3* folder, not within the directory.

2. Transfer *Prog3.tgz* to the Program 3 submission area of CourseSite.

Comment Block:

```
/*
    CSE 109: Fall 2017
    <Your Name>
    <Your user id (Email ID)>
    <Program Description>
    Program #3
*/
```