Homework 06 - Singly Linked List and malloc/free Interposition

Due Wednesday 10/24/2018 11:59pm

What to get before you start

Download hw6handout.tar file from Canvas. It will include the following files.

```
hw6handout/list.c
hw6handout/list.h
hw6handout/main.c
hw6handout/Makefile
hw6handout/mymalloc.c
```

What to do

1. Modify Makefile, so that besides the ./main program your Makefile will also create **./mainlink** program that uses **link time** interposition method to overwrite library function calls to malloc and free. When you run the program ./mainlink, every function call to malloc or free in the program will print out a trace. Review slides and code from Chapter 7 (week07.tar) for details. The link time interposition uses the linker's --wrap option to resolve any calls to malloc now to __wrap_malloc and resolve any calls to __real_malloc to the actually malloc call in the library.

2. The main body of the list implementation is already implemented for you, which includes push/pop/display/destroy and the engine to manipulate a list. All you need to do is to Implement the following four functions in list.c file.
   a. `int count(listNode *listPtr, int value);`
   b. `void insert(listNode **listPtr, int value);`
   c. `void reverse(listNode **listPtr);`
   d. `void removeAll(listNode **listPtr, int value);`

Here is some details about each function's implementation.

a. `int count(listNode *listPtr, int value);`
   Count how many times the value appeared in the list. This function should not need to allocate or free any memory.
b. `void insert(listNode **listPtr, int value);`
   Add a new node with value to the TAIL of the list. This function should call malloc exactly once. If the original list was empty, remember to update the head pointer stored in location pointed to by listPtr.
c. `void reverse(listNode **listPtr);`
   Reverse the order of all nodes in the list. Do not allocate or free any memory space for this function. You should be able to reverse the list simply by relinking/rearranging the existing allocated nodes.
d. `void removeAll(listNode **listPtr, int value);`
   Remove all nodes that has the given value in the list and free the memory space

for those nodes. Please make sure that after removing the value, the remaining list is still properly linked together. Also watch out for the corner cases of the value is at the head or tail of the list.

More details about suggested traces of execution to test your code will be posted on Piazza. Please make sure you get notification from Piazza properly.
You will only need to make changes to **Makefile** and **list.c** file. DO NOT Modify any other files in the handout folder or add or remove any files or change any files' names in the folder. When we test your code, we will only copy over your Makefile and list.c file to see if it works with the rest of the program provided to you.

What to submit
Submit a tar file hw6handout.tar that contains the same set of files as hw6handout.tar you initially get, with only **list.c/Makefile** modified to complete the homework.