Prelab 9 for March 22nd

For this prelab you will implement the following list functions using a simple linked list. Every function except deleteList should take only a constant amount of time, i.e., not proportional to the length of the list. Returned error codes should be either 0 = *no error* or 1 = *error*.

```
/* This function returns an empty list. Parameter is a reference to an error code. */
List initEmptyList(int *)

/* This function inserts the pointer to a user's object at the head of the list.
   Returns error code. */
int insertHead(void *, List)

/* This function returns the object at the head of the list. NULL is returned if
   the list is empty. */
void * getHeadObject(List)

/* This function removes the object at the head of the list. Returns error code. */
int removeHead(List)

/* This function inserts the pointer to a user's object at the tail of the list.
   Returns error code. */
int insertTail(void *, List)

/* This function returns the object at the tail of the list. NULL is returned if
   the list is empty. */
void * getTailObject(List)

/* This function returns the number of objects in the list. */
int getLength(List)

/* This function deletes list, and returns an error code.
   No operations can be performed on deleted list */
int deleteList(List)
```

This prelab is designed to reinforce your understanding of three critical concepts. The first is how void pointers can be used to accommodate user objects of any data type. In other words, these interface functions can be used generically to store objects of any kind. The second concept, which is much more subtle, involves completely separating the ADT datatype provided to the user from the structs used for the underlying implementation. The third concept is the specification of a performance constraint that every interface function except deleteList is performed in a constant amount of time.

Note there is no need to use dummy/placeholder nodes, though you should understand how to use them. I recommend implementing with and without them to see how they can significantly simplify your code in some cases by eliminating the need to test for special cases. I also recommend using them because their use may be required in the next or future lab.

This is arguably the most important prelab of the semester.