Prelab 7 (for October 21st)

For this prelab you are to create a List ADT that includes the following interface functions:

```
/* These functions insert the object of the first parameter
   at the head/tail of the list and returns an error code:
   0 = success, 1 means there was insufficient memory
   and the list is not changed. */
int insertAtHead(void*, List)
int insertAtTail(void*, List)

/* These functions delete and return the object at
   the head/tail. If list is empty, NULL is returned. */
void * removeHead(List)
void * removeTail(List)
```

Note that a List object is being passed – *not a pointer*, i.e., not as **List** *, which means the user must declare their list object, e.g., a List variable emplist, as **List empList** rather than **List * empList**. In other words, your List ADT behaves *exactly* like builtin datatypes. How can that be achieved?

Consider the following:

```
typedef struct{
   Node *head;
} List;
```

Now the real list pointer is "hidden" in a struct so that users can declare their variables using what looks like an ordinary built-in data type. You might be thinking, "*Okay, that makes sense, the List struct is just a kind of wrapper for the real list pointer.*" That's right, but to avoid forcing the user to pass it by reference, we'll need to ensure that the List struct never changes after the user calls the init function to create it. Hmm, how can we do that? Could it involve... *a dummy node*?

I should emphasize that documentation for your interface functions should be significantly more complete than what I've given. Put yourself in the place of the user and try to provide information that is as clear and complete as possible. The following is a bit better:

```
/* This function inserts an object at the tail of a list.
   First parameter is a pointer to the object to be inserted.
   Second parameter is a List.
   Pre-condition: The value of the error code is undefined.
   Post-condition: The error code will have a value of 0
        if the insertion was successful or 1 if there was
        insufficient memory. The list will be unchanged if
        the error code is nonzero.  */
int insertAtTail(void*, List)
```

- JKU

P.S. Although I didn't mention it, you'll of course need to provide **initList** and **freeList** functions. Because I haven't specified their prototypes, you'll have to decide them for yourself.