Recitation 8: Implementing a Linked List

COS235

April 7, 2015

1 Programming Task

In this lab you will be implementing a few functions to complete a partial linked list implementation. In this implementation, A `List` is created which points to a `ListElement`. A `ListElement` then can point to the next `ListElement` in the list or `NULL` if the element is at the end of this list. `ListElements` should be created dynamically using `malloc` to allocate memory and removed using free to release the memory held by a `ListElement`.

The implementation is broken up into set of generic functions, that will work for a list containing any type of data, and a set of type-specific functions, that are implemented for the desired type of data to be stored in a `ListElement`. These type-specific functions should be implemented using the generic functions. See the implementation of the function `addData` as an example. Using abstraction like this allows future changes to be localized and code to be more easily reused. What would need to change if you renamed a member (field) of a `ListElement`? What if `listData` were a pointer to an int? What if you switched to a doubly-linked list where each element had a pointer to the next and preceding element? These changes can all be made through small localized edits.

You will need to implement the functions listed below. It may be helpful to draw out a list and think about conceptually what needs to be done before you begin programming.

- `listSize` returns the number of elements in the list
- `insertElement` inserts an element at a specified index
- `removeElement` removes the element at the specified index
- `getData` get the data stored in the element at the specified index
- `insertData` creates and element with the desired data and inserts it at the specified index in the list
- `removeData` removes the element at the desired index and returns the data it holds
- `sort` optional method to return a sorted copy of a supplied list

The source files you will be working with are:

- `list.h` a file containing the definition of the `List` and `ListElement` struct and type and the functions to be implemented.
- `list.c` a file containing the implementation of the functions defined in `list.h`.

2 Compiling

```
gcc -ansi -pedantic -Wall list.c
```

3 README

For programming assignment for this course you should include a README file, either with no extension or .txt, that includes the following information:

- Your name
- The date
- The course and assignment
- Sources you used
- People you worked with and what you did together
- A listing of the included files
- Instructions for running your program, including commands

If you are submitting a lab assignment done in a group, then only one submission is necessary and all of your names may be listed in one README file.

4 Deliverables

A zip or tar.gz file containing:

- .c and .h files
- README file
- Example output
- Makefile (optional)

This only needs to be submitted one by member of a group.