CS 241L - Data Organization

Fall 2022

Programming Assignment #6
Linked Lists functions

Total points: 100

Due date: October 29^{th}, 2022

A linked list is a linear data structure that allows us to add and remove items from the list very quickly, by simply changing a few pointers. There are many different variations of linked lists. We have studied the *doubly-linked, circular, with a dummy-header-node* version of a linked list. In the `linkedList.c` program supplied on Week 9 and in the lectures that week, we studied several functions to manipulate a Linked List.

For this assignment you must write the code for the following additional linked list functions: `addFirst, removeLast, clear,` and `set`. The provided source code contains a stub for each of these functions, together with a description of what the function should do. You must complete the body of each of these functions. You must write each of these functions from scratch. You must not invoke the existing list functions.

The `addFirst` function will insert a new data item into the list at position 0 (in front of all the items already in the list).

The `removeLast` function deletes the last item in the list (at position size-1), and returns the data that was deleted from the List. If `removeLast` is invoked on an empty List, then your program should terminate processing and exit with a status of 2.

The effect of the `clear` function is to make the List become an empty List.

The `set` function should alter the value of the data at the specified location in the list. The size of the list remains the same. The `set` function should return the item that was deleted from the List (because it was overwritten by the new item). If the `set` function references an illegal index, then your program should terminate processing and exit with a status of 2.

You should implement each of these functions in the **most efficient manner possible**. Your function should not take O(n) time when an O(1) solution is possible. Also, the `set` function should

be as efficient as possible. If the target location (e.g., index, position) is more than half-way down the linked list, then the traversal should start from the rear of the list and move leftwards.

1 Assignment

In the place where this statement is located you will find a C program called: `pa6_LLF.c`. You should modify this code to add the body of the functions that will be implemented by you.

The main function of your program should read in a sequence of list operations from stdin. Each line of the input file will specify one list operation. The first char of the line will indicate a specific list operation using the following definitions:

```
'a'  Add the given key to the end of the list (i.e., append)
'f'  Add the given key to the front of the list (i.e., position 0)
'd'  Delete from a specific location in the list
'r'  Remove the last key from the list
'o'  Output the list to stdout
'c'  Clear the list
's'  Alter the specified position in the list to contain a new value.
```

Note that each string stored in the list will contain at most 20 letters. The given source code already contains the functions for 'a', 'd', and 'o'. You must not alter these functions. For example, for the input file below:

```
a alice
a bob
a chad
d 1
f abby
s 2 ed
a fran
s 1 bill
r
o
c
o
Your program should produce the output:
[abby bill ed]
[]
```

Two input files and their corresponding output files are provided in the same place in canvas where the .c template program and this document are. There will be duplicates of all the necessary files in the cs machines, the full directory path to those files is: `/nfs/faculty/soraya/cs241_fall2022/PA6`

Two input files and their corresponding output files are provided in the tarball: `programming_assignment_6.tar.gz`. To extract these files you should use the command

```
tar -xvf programming_assignment_6.tar.gz
```

This will create a folder called `extra_files` containing the input and output files mentioned above.

If the syntax of the input file is faulty, then your program should return an exit code of 1. For example, if the 'a' or 'f' operation is not followed by a string on the same line, or if the 'd' operation is not followed by an integer on the same line. If the requested list operation has valid syntax, but the requested operation is invalid, such as 'r' on an empty list, or an attempt to 's' a position in the list that does not exist, then your program should immediately return an exit code of 2.

The week 9 module on canvas on Lecture #16 contains code for the linked list data structures and functions that we covered during lectures, reviewing those will prove helpful for this assignment.

2 What to submit

You should be sure to include THE NAME OF THE AUTHOR OF THE PROGRAM in a comment at the top of your source code file, for this and all other assignments in this course.

Your source code must use proper style, that is variables should be well named (name is not too short, not too long, and is meaningful), and bodies of loops, if's, etc. should be properly indented. Refer to the coding style file for this class, published on Canvas under Coding Standards in a file named: `cs241_codingStandards2020.pdf`.

When you copy the `pa6_LLF.c` file for this assignment to modify it according to these instructions, rename it using your last name and the initial of your first name, like this: `lastName_initialFirstName_LLF.c`

Submit the file `lastName_initialFirstName_LLF.c` for grading to Canvas in the place of this assignment.

3 Grading rubric:

If any of your C programs/functions does not compile with the `-std=c99 -pedantic -Wall` options without errors or warnings the points given for the assignment will be zero. Otherwise, the following rubric will be used:

+ 10 pts: Your C files follows the class coding style.

+ 15 pts: Your program process correctly the input data from `stdin`.

+ 55 pts: Your file provides implementation of the functions `addFirst, removeLast, clear`, and `set` meeting the aforementioned specification.

+ 5 pts: after processing the instructions from the std in, the program frees the memory claimed by previous calls to malloc; this is independent of the input files containing a call to the `clear` function. In other words, you must make sure that your program does not leak memory.

+ 10 pts: Your program matches the output of the input files provided to you in the tarball.

+ 5 pts: Your program matches the output of internal files only available to the grader.