CSCI 2720 Data Structures
Spring 2013

Project 1 - Traversal of Linked Lists

Assigned: Thursday, Jan 31st Due: Friday, Feb 15th 11:59pm

Project Description

Introduction

In this project, your task is to construct and maintain a linked list where each cell contains a search key associated with a nonnegative integer value. The program must work iteratively with the user and perform the operations specified by the user.

Your program must support three different types of operations including Insertion, Deletion and Search. Insertion requires your program to insert a specified search key and its associated integer to the linked list. If the search key is not found in the linked list, the program must generate *a new cell on the head of the linked list* to store the search key and the integer, otherwise, the integer associated with the search key is replaced by the one specified by the user. The Deletion operation removes the cell that contains the search key provided by the user from the linked list and does nothing if none of the cells contain the search key value. When subject to the Search operation, the linked list must be exhaustively traversed to look for the cell with the provided search key and ***output** the associated integer*. In addition, your program also needs to locate the preceding cell with a distance of the obtained integer to the cell that contains the key specified by the user and ***output** its search key* (see Ch. 3.4 in the textbook and exercise 12).

No duplicate search keys are allowed in the linked list. In other words, when insertion is performed, the linked list must be searched to ensure that the inserted search key does NOT exist in the linked list before a new cell is generated.

Input format

The input format for different operations are specified as follows:

- Insertion operation: I [Search key] [Integer]
- Deletion operation: D [Search key]
- Search operation: S [Search key]

Implementation

You are required to implement a class to generate and manipulate the linked list. Operations must be implemented with different functions in the class, and you are required to implement the Search operation with **either one** of the following approaches discussed in the class:

- Two pointers that move only forward in the list (similar to algorithm 3.9 in the textbook)
- A "zig-zag" scan with a stack of pointers

You may assume that the search key is of string type. You may choose to implement any of the above two algorithms for your search and **must** mention which one you picked in your Readme file.

You don't need to implement your own stack but use Stack from the Standard Template Library (if you choose to implement the second algorithm - "zig-zag" scan with stack).

Test Programs
Your program will be tested and graded based on the performance on the following input files, full credit will be given if your program is written in a good style and works perfectly well on them.

1. Input1.txt
2. Input2.txt
3. Input3.txt

Example Run

```
I  element1 1
element1-1
I  element2 2
element2-2, element1-1
I  element2 3
element2-3, element1-1
I  element3 3
element3-3, element2-3, element1-1
I  element7 2
element7-2, element3-3, element2-3, element1-1
D  element3
element7-2, element2-3, element1-1
D  element5
element7-2, element2-3, element1-1
S  element7
2
S  element1
1  element2
```

Explanation for the first search: element7 is at the head of the linked list, going back 2 positions from there is not possible. Therefore we only print the associated integer.
Explanation for the second search: element1 is located as the third node and it is associated with integer 1, going back 1 node from element1 returns element2. We print element1's associated integer and the key of the previous node: element2.

Submission Procedure

All submissions are to be done through eLC. You can attach several files separately while submitting through eLC so you don't have to zip/tar your source files. If you have any problems using eLC submission, please inform us before the deadline so that we can resolve the issues. It is your responsibility to make sure that your code compiles and runs on **nike** before submitting. Please submit source files and source files only, no executables.

Non-compiling submissions will result in a grade of 0.

As stated in the Syllabus, you can return it up to three days late, receiving 20% off of your possible grade for the project (10% for the first day, 5% for the second and third). You cannot submit your project after 3 days pass due date.

You should submit a Readme file that specifies which algorithm you have implemented and includes instructions on how to compile and run your code. In addition list.h, list.cpp, node.h and node.cpp (and any other files you deem necessary), you should also submit your makefile with targets to compile and run your program.

You will be given skeleton files, `list.h and node.h` to start with your project.

Grading on Style

Grading will be based on correctness and completeness, as well as style. Elements of style include:

- Code should be clear (structure) and easy to read (formatting).
- Code should be well commented.

Grading Rubric

Deliverable	**Total Points**	**Points**
Documentation/Readme	10	
Source Code	80	
main	15	
Insert	15	
Delete	15	
Print	5	
Search	30	
makefile	10	