Florida Polytechnic University – Data Structures and Algorithms
Programming Assignment 1 Spring 2023 – Linked List

PROGRAMMING ASSIGMENT 1: Single Linked List

Submission deadline: February 2nd at 11:59 pm

Deliveries:

A cpp program (the source code, not the executable).

Requirements: The program should run using the g++ compiler of university UNIX system (ember).

Note: Use your student ID number as your program name.

Note: Note: You can start your implementation in any C++ programming environment such Visual Studio or Google Colab. However, you need to test your program in the university ember system.

I certify that I coded this program by myself, and this code doesn't correspond to the intellectual work of someone else.
Signature: _____
Add this document with your signature when you upload your homework. No doing that will result in an automatic ZERO.

Rubric

Implement the 13 functions of the API for ***Single*** **Linked List**. Each correct implemented and tested function has a value of 7.7 points.

Description

You should implement the API for single Linked list described in Figure 1. I am giving you three different implementations you can use for guidance.

Here, a nice blog which explains step by step the elements of a linked list and provide code

Implementation 1: Use as guidance if you want.

Blog: https://www.codementor.io/@codementorteam/a-comprehensive-guide-to-implementation-of-singly-linked-list-using-c_plus_plus-ondlm5azr

GitHub account associated with blog: https://github.com/kamal-choudhary/singly-linked-list/blob/master/Linked%20List.cpp

This implementation create a *tail* pointer at the end of the list

Implementation 2: Use as guidance if you want.

Here, another implementation: easy to follow (slopy programming). Here, the link: https://gist.github.com/charlierm/5691020

Implementation 2: Use as guidance if you want.

This implementation is inspired by the first and second implementation, so it is formal and intuitive. Here, the Link: https://gist.github.com/csaybar/d0451939e79c16456095d6ed59bd718f#file-single_linked_list-cpp-L3

My advice

1. Download the implementation of your preference, move it to the ember system using SecureShell. Open it, and compile it. You will be using it to see how they do things. You can recycle some of these code if consider helpful.
2. Now start your own implementation, it is also better start with your own implementation that use code of someone else, believe, you have control of your own stuff.

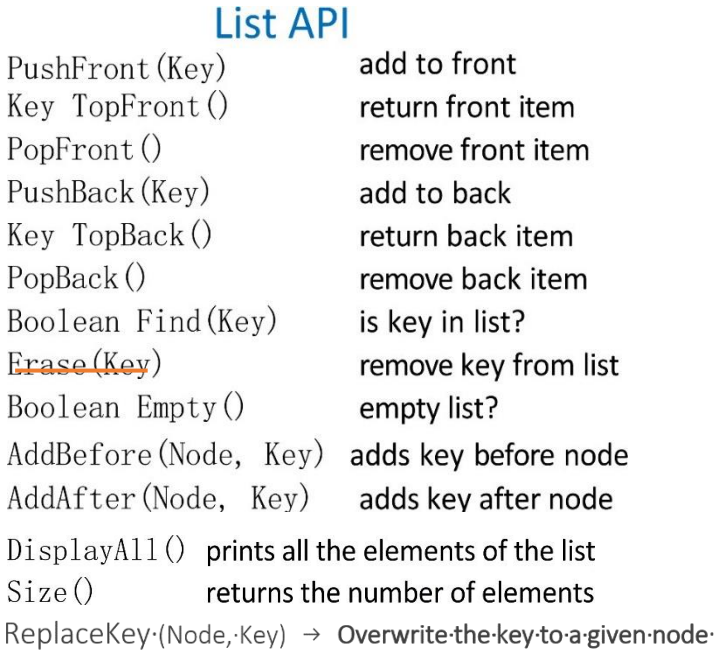

Figure 1.

Implementation details

1. Use constructors to initialize the objects.
2. Some functions should handle the case of an initial empty list and if it necessary create first node. For instance push front() if the list is emptly should create the firs node.
3. Don't use a function create_node() to populate the List, you can use something similar internally, but don't let this function available to the user.
4. Populate your list using the functions PushFront(key) and PushBack(key) All the functions should handle the case of an empty List.
5. As you create any node keep a tail pointer at the last element of the list (check how the provided implementation do that, you can take ideas from them)

Testing

Populate the list executing the following functions of the API

PushFront(25)
PushFront(50)
PushFront(90)
PushFront(40)
PushBack(35)

DisplayAll()

40 90 50 25 35

TopFront() -> 40

TopBack() -> 35

popFront()

DisplayAll()

90 50 25 35

popBack()

90 50 25

DisplayAll()

90 50 25

PushBack(10)
PushBack(12)
PushBack(14)

DisplayAll()

90 50 25 10 12 14

FindKey(25) ->TRUE

FindKey(6)->FALSE

Empty() -> FALSE

Let’s assume that the first element is in position 0.

Then execute

AddBefore(3, 94)

DisplayAll()

90 50 25 94 10 12 14

AddAfter(2, 5)

DisplayAll()

90 50 25 5 94 10 12 14

Size() ->8

ReplaceKey (6, 87)

DisplayAll()

90 50 25 5 94 10 87 14

PopFront()
PopFront()
PopFront()
PopFront()
PopFront()
PopFront()
PopFront()
PopFront()

Empty() -> TRUE