**Students:**

This content is controlled by your instructor, and is not zyBooks content. Direct questions or concerns about this content to your instructor. If you have any technical issues with the zyLab submission system, use the **Trouble with lab** button at the bottom of the lab.

# 3.24 Lab 1: List and List Iterator Classes (Summer B 21)

## Due Thursday, July 29 (demo in lab 2) - Submit via Canvas by 11:59 PM

### Summary

In this lab you will gain practical experience working with a List class and List Iterator class. An iterator abstraction is commonly used to provide a mechanism to visit individual items in a container. Many of the The C++ Standard Library container classes provide this. For example the list class (which is a singly-linked list implementation ) does. Understanding what an iterator is and how it functions should give valuable insight when you want to use an off-the-shelf data structure from a library. Specifically you will extend the functionality of pre-existing source code provided in the framework (list_source.tgz). Students will implement various list insertion member functions as well as a reverse print function. Make use of the examples for Iterator use in the list print() function to extend the Iterator class to be able to traverse a list in both the forward and reverse direction. Two question you should consider are: Is the List class a singly-linked list or doubly-linked list? Does it matter for the implementation of a reverse capability for the Iterator?

Use of Cloud9 Requirement: Cloud9, the browser-based IDE (Integrated Development Environment) that the CSE department has supplied for you to implement your programming tasks in this course is REQUIRED. Students who fail to use AWS and Cloud 9 in the course will be docked points on each lab or assignment they complete until they do make the switch. This will be true no matter what alternate IDE they may use currently or reason given for ignoring course directions. Lastly, for full credit, you must submit a zip archive (name it lab1.zip) that contains your solution source code via Canvas (Assignments->Lab 1) and demo working code for TA. This will be the procedure for all labs this quarter.

Instructions for getting an AWS account and setting up Cloud9 are in the Lab 1 link on Canvas. The file is named "CS010C_AWS_Account_Setup_Instructions_sum21.pdf." The instructor will walk you through this process at the beginning of lab 1.

- Iterator.h
- List.cpp
- List.h
- main.cpp
- Node.cpp
- Node.h

## List, Iterator Classes

The *List* class is a *doubly-linked* list with an associated *Iterator* class. The List class currently has working member functions: **push_back(), push_front(), and print()** and a **head** pointer. The Iterator class has a public interface of read (get) and write (set) member functions. The *Node* class has private data members: **Node* next, Node* prev, int value** with a public interface of read (get) and write (set) member functions is provided. Read-only member functions commonly use *get* in their function names and are referred to as *accessor* functions. Write member functions commonly use **set** in their function names and are referred to as *mutator* functions.

The **OOD/OOP (Object Oriented Design/Object Oriented Programming)** principle of *data encapsulation* is strictly enforced in the provided source code. All read-writes of internal data members (*private*) of a particular class **must** occur via that class's public class member function interface. **DO NOT** make the List class, or Iterator class a friend of the Node class in order to allow direct manipulation of the Node class's private data members.

Follow the steps below obtain the archived source code and get started. (The '$' denotes the start of the command line prompt.).

0. Download the source code to local machine. (list_source.tgz) is available from the Lab 1 link on Canvas.
1. Create an AWS (Amazon Web Service) EC2 instance (follow directions in AWS sign-up slides), named **netid*cs010*sum10a**. The **netid** is your UCR R'Mail alias, e.g. *kflynn001*.
2. Create a **labs** folder in the root directory **netid*cs010*sum10a** in Cloud9.
3. Create a **lab1** folder in the "**labs** folder.
4. Right-click (or touch down and hold) on **lab1** folder in the Cloud9 file explorer.
5. Select File->Upload Local Files from the pop-up menu.
6. Drag & Drop list_source.tgz to the "Upload Files" window.
7. Right-click on **lab1** folder and select *Open Terminal Here*. This will open a terminal with a command prompt **$** in the **lab1** folder.
8. Unzip and extract source files. Type the following command into the terminal window

```
 * CS 10C (RR)
 * Summer 2021
 * Lab 1
 * Iterator.h
 */

#ifndef ITERATOR_H_
#define ITERATOR_H_

#include "Node.h"

class Iterator
{
public:
    Iterator();
    ~Iterator();
    int get_value()const;
    Node* get_curr_pos()const;
    Node* get_last_pos()const;
    void set_curr_pos(Node*);
    void set_last_pos(Node*);
    bool is_equal(Iterator rhs)const;
    void next_pos();

    /* You must implement this ! */
    void prev_pos();
    void operator++();
    void operator--();
    bool operator==(Iterator rhs) const;
    bool operator!=(Iterator rhs) const;

private:
```

```
List contents:
0, 1, 1,

List contents:
0, 1, 1, 2,

List contents:
0, 1, 1, 2, 3,

List contents:
0, 1, 1, 2, 3, 5,

List contents:
0, 1, 1, 2, 3, 5, 8,

List contents:
0, 1, 1, 2, 3, 5, 8, 13,

List contents:
0, 1, 1, 2, 3, 5, 8, 13, 21,

List contents:
0, 1, 1, 2, 3, 5, 8, 13, 21, 34,

List contents:
0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55,
```

```
8, 5, 3, 2, 1, 1, 0,

List contents:
13, 8, 5, 3, 2, 1, 1, 0,

List contents:
21, 13, 8, 5, 3, 2, 1, 1, 0,

List contents:
34, 21, 13, 8, 5, 3, 2, 1, 1, 0,

List contents:
55, 34, 21, 13, 8, 5, 3, 2, 1, 1, 0,

List Destructor called... Deleting.... 55
Deleting.... 34
Deleting.... 21
Deleting.... 13
Deleting.... 8
Deleting.... 5
Deleting.... 3
Deleting.... 2
```

called on each list instance *myList* and *myList2*, debugging information for each node is outputted to the console as the node's memory is returned (deallocated) to the *heap* aka *free-store*.

## Additional Required Functionality:

- void List::sorted_insert(int value) - functions inserts new node with value into list. This insert_sort should preserve the increasing value ordering for the list. For example, if value == 5, then the new node would have a value of 5 and be inserted after the last instance (furthest from head node) of an existing node with value of 5 or less. To test create a list object with sorted values. NOTE: When inserting at the front or back of list, you can call push*front and push*back in the body of the sorted*insert function. DO NOT, however, call push*back or push_front directly in main function.
- void List::print_reverse() const - function prints the contents of a list backwards. NOTE: Implement a void Iterator::prev_pos() function to make for use in this function. Assume that the list must be sorted before insertion and after the operation completes to be

Every file that you create or edit in the course must have the contents of **class_file_header.txt** filled in by YOU, at the top of the file. Download this file from the Lab 1 link on Canvas.

## Canvas Submission:

Steps 1-6 only need to be completed once, before student submits lab1.zip file to Canvas.

- **Change the default download archive format to zip.**

1. In AWS Cloud9 right-click (or touch down and hold) on black-and-white *Cloud9 logo* in upper-left corner of browser

```
session)
15/15 pts turn-in code submitted in zip on Canvas and compiles

Functionality
--------------------
```