CS014 - Lab 1 - Linked List

Prof: Ryan Rusich rusichr@cs.ucr.edu **TA:** Joshua Frear jfrea001@ucr.edu

Synopsis: In this tutorial you will be using pre-existing source code from an archived file using any prefered text editor or IDE. Once you are able to edit the files, compile and run it successfully, you will add new functionality to the project via your own original source code additions.

The domain of the project will be a most fundamental data structure, a doubly-linked, linked list. The source code provided to you (see below) supports push back, push front, and printing the list. In addition, this implementation makes use of a user defined Iterator object for traversing the list and accessing its contents. One caveat of the implementation is that it does not make any classes friends of other classes, but rather adheres strictly to OOD/OOP (Object Oriented Design/Object Oriented Programming) principles, i.e. data encapsulation.

Getting Started: First you will untar the source code. Use the following steps and commands to do so. The $ signals the start of the command line. Do not enter the $ or the space that follows it. Also do not include the < > in the username placeholders.

Step 1 (Linux/OSX): Remotely log into bolt via SSH. SSH clients come pre-installed on MacOSX and popular Linux distros. Open a terminal and enter the command and put in your password when prompted:

```
$ ssh <cs_username>@bolt.cs.ucr.edu
```

Step 1 (Windows): Remotely log into bolt via SSH using PuTTy. Download PuTTy, put the executable someplace you can get it easily (the desktop and/or the taskbar), run the executable and use the following info:

Host Name: <cs_username>@bolt.cs.ucr.edu
Port: 22

In Saved Sessions box, type *UCR Bolt* and click “Save”. Now in the future you can just double click *UCR Bolt* and it will run with the same Host Name and any settings you change.

Personal Opinion: Before clicking save, click “Colours” in the side menu, then look for ANSI Blue and modify it to a lighter shade of blue, the default shade is very dark on some monitors. Click “Session” in the side menu. Then click save. This setting will be saved for future logins.

Enter your CS password when prompted.

Step 2 (Linux/OSX): Download the source code from iLearn. You will be placing it in the home directory on your bolt account. I am assuming the file is located in the Downloads directory which is normally in your home directory (the ~/ path). If you placed it somewhere else, be sure to substitute the path in the command below. Open a second terminal (do not use your SSH terminal, keep that open elsewhere). Enter the following command to transfer the source code FROM your local machine TO bolt. Enter your cs password when prompted:

```
$ scp ~/Downloads/list_source.tgz <cs_username>@bolt.cs.ucr.edu:~/
```

Close this second (non-SSH) terminal. Return to the first terminal window for the remaining steps.

When it comes time to submit the work, you will enter a similar command to transfer your completed tarball FROM bolt TO your local computer so you can access it in your web browser. Don't enter this command now. Be sure you enter this command in a newly opened, non-SSH terminal just as you did to transfer the source code to bolt.

```
$ scp <cs_username>@bolt.cs.ucr.edu:~/Lab1.tgz ~/Documents/
```

Step 2 (Windows/OSX): Download the source code from iLearn. We will be placing it in the home directory on your bolt account. I recommend downloading and installing CyberDuck. Run CyberDuck, click "Open Connection". Click the top most drop down box that should have "FTP…" as the contents and select "SFTP (SSH File Transfer Protocol)". Enter bolt.cs.ucr.edu for the Server and enter your CS account credentials. Leave port on 22. If this is your personal machine I recommend saving the password. Click connect. The rest is a simple drag and drop interface. Be sure you place the list_source.tgz in your home directory only. Placing it in an incorrect location will cause errors in the remaining steps. After you are done, return to your SSH terminal/PuTTy window.

You can use CyberDuck to retrieve your completed tarballs FROM bolt when it is time to submit to iLearn.

Step 3: Make a directory on bolt called Lab_1:

```
$ mkdir ~/Lab_1
```

Step 4: Make sure you are now working in the Lab_1 directory:

```
$ cd ~/Lab_1
```

Step 5: Untar the tar ball, putting all the source code into the current Lab_1 directory:

```
$ tar -zxvf ~/list_source.tgz
```

Current Program Behavior: The main function creates two Lists called myList and myList2. It first attempts to print an empty list. Next it pushes back the first 11 Fibonacci numbers into myList, which are obtained from a local recursive function inside the main.cpp. The contents of myList are printed to the console iteratively with each successful push_back. Next, myList2 is filled with the contents of myList using the List member function push_front, effectively reversing the contents of MyList in myList2. Once again, the contents of myList2 are printed to the console iteratively with each successful push_front. Last, you will see some debugging information displayed to the console from the List class destructor showing the value of nodes as they are destroyed.

To run the code, first compile your program. Ensure you are in the correct directory (see Step 5 above), then run the following command in the terminal:

```
$ g++ -o lab_1.out main.cpp List.cpp Node.cpp Iterator.cpp
```

To run the program you just complied:

```
$ ./lab_1.out
```

When you successfully run the program that was provided in list_source.tgz, the output to the terminal should be :

```
Test printing an empty list:
List contents:
empty

Pushing back first 11 Fibonacci numbers:
List contents:
0,

List contents:
0, 1,

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,

Reverse myList by pushing front, contents of myList into myList2:
List contents:
0,

List contents:
1, 0,

List contents:
1, 1, 0,

List contents:
2, 1, 1, 0,

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

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

List contents:
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
Deleting.... 1
Deleting.... 1
Deleting.... 0

List Destructor called...
Deleting.... 0
Deleting.... 1
Deleting.... 1
Deleting.... 2
Deleting.... 3
Deleting.... 5
Deleting.... 8
Deleting.... 13
Deleting.... 21
Deleting.... 34
Deleting.... 55
```

Editing the code

Since we are using an SSH connection we are slightly limited in how we may edit the source code. You are free to edit the file however you like, but I personally recommend learning how to use vim. It is basic, but can be used directly in the SSH terminal. Here's a sample tutorial (Sections 1 and 2 are the bare minimum to use vim). Emacs and vi are also good alternatives. These in-terminal editors will be especially useful in future courses and can be used to edit all text-based files.

If you want a GUI program, but still want to be able to work directly from bolt, search online for X11 solutions. This will let you use GUI's over SSH. There are multiple X11 solutions for all major OSs. This will require research and setup on your own.

If you chose to work offline on your own machine, you will have to transfer your files between your local machine and bolt through the methods described in Step 2 above when you want to run your code.

You MUST compile and run on bolt for demos. All graded code will be tested on bolt. If your code doesn't compile on bolt your will receive no credit. If your code misbehaves on bolt, but works fine on your personal computer, only the results produced on bolt will be graded.

Extending the Linked List

1. Write a function *void List::sorted_insert(int value)* that takes an integer value inserts it into the list as a new node in the list. This insert_sort should preserve the increasing ordering of the list, i.e. if you were to insert a 5, then the new node would have a value of 5 and be inserted after the highest integer less than 5 and before any integer equal to or greater than 5. You can assume that the list is already sorted, though might be empty or have one element. Place a restriction that push_front and push_back can only be used as subroutines for sorted_insert when operating on a single list, i.e. you cannot push_back or push_front directly(in main) into a list that you intend to have as sorted.

2. Write a *void List::print_reverse()const* function that prints the contents of a list backwards. You might find it useful to implement a void Iterator::prev_pos() function to make this more straightforward.

3. Write a *void List::insertAfter(int i, int value)* function that inserts a node with specified value at position i+1 in the list. If this position is outside of the list's current boundaries, be sure to account for this with a sensible solution...

4. Write a *void List::insertBefore(int i, int value)* function that inserts a node with specified value at position i-1 in the list. If this position is outside of the list's current boundaries, be sure to account for this with a sensible solution...

5. Write a *int List::count(int value)const* function that returns the number of times *value* appears in the list.

Submitting Code to iLearn

Step 1: Pack up all the files into a tar. Add any additional files you might've added that are needed to run the code. Do NOT include non-source code file such as the .out, .o, or other files generated from compiling your code. Run the following command while SSH'ed into bolt

```
$ tar -zcvf ~/Lab1.tgz main.cpp Iterator.cpp Iterator.h List.cpp
List.h Node.cpp Node.h
```

Step 2: Transfer the file to your local machine from bolt. Run the following command while SSH'ed into bolt. Change the ~/Documents directory to wherever you want the tarball on your local machine:

```
$ scp <cs_username>@bolt.cs.ucr.edu:~/Lab1.tgz ~/Documents/
```

If you're using CyberDuck, then instead log in through CyberDuck, go to the home directory. Drag and drop the .tgz file somewhere on your local machine.

Step 3: Log into iLearn and submit your Lab.tgz under Lab 1. You are allowed unlimited submissions.

Step 4: Don't forget to demo your project to your TA.