# CSC 205 Lab 10 : Linked Lists

## Goals

After completing this lab, you should be able to:

* Understand object references and the advantages of a dynamic linked list over an array.
* Be able to display and count the contents of a linked list.
* Insert a node into a specified position of a linked list.

## Lab Startup

Change into your Labs directory, and let's create and change into a Lab10 directory.

Now, let's copy over some files by typing : cp /pub/digh/CSC205/Lab10/\* .

**Building and Tracing a Simple Linked List**

Take a look at the Links.java program in your attached handout. Draw the linked list that would be created by the main method.

Now, compile and run the Links program and check to see if your list prints out in the manner that you would expect given a head pointer to the front of your list.

**Writing Simple Methods for Processing Linked Lists**

Add a recursive method named count to your Links program that can be used to count and return the number of nodes pointed to by a head pointer. Your method will have one parameter, the pointer head.

Add an iterative method named findMax to your Links program that returns the largest value in a linked list pointed to by a head pointer. Your method will have one parameter, the pointer head. You should not be concerned about the data type of the elements stored in your nodes. Use the Comparable interface appropriately in your method.

Include method calls in main to test both of your new methods.

**Building an Ordered Linked List**

##### Draw the linked list on a separate sheet of paper that will be created by the main method in your Links2 program shown on your handout. Trace through the insert method very carefully and show your prev and curr pointers. Notice that very powerful for loop with an empty body that uses multiple initial conditions, dual boolean conditions, and multiple increments. Using the comma in this manner is perfectly legal in Java for loops.

##### Compile and test the Links2 program to test your results when you’re done.

**Searching an Ordered Linked List**

Take a look at the Search program in your current directory. You need to add two methods to this program so that it will successfully read in 0 or more integers from the keyboard, insert each of them into a sorted linked list, and then allow the user to search for a key from your list.

First off, in buildList, you will need to declare a local head pointer and flush it to null. Next, use the Scanner class to read in one or more tokens from the keyboard.

You’re now ready to continue reading in tokens until the end of file is reached. To simulate end of file from the keyboard, you’ll use Control-D. Each token that is a number should be inserted into the linked list using the insert method. Be sure you assign the result of this method to head each time in case your new item was added to the beginning of your linked list or your list was initially empty.

Finally, complete the search method that takes the head pointer and a key value, and returns true or false indicating whether your key value is present in your list. You may use iteration or recursion.

Test your program using the sample run below where user input is shown in boldface.

**Sample Run**

Please input 0 or more values at keyboard

**12 4**

**-1 5 3**

**0**

**2**

Now printing list

-1 0 2 3 4 5 12

What key in list are you searching for? **15**

Your key was not found.**Lab 9 Program for Tracing**

The Links Program

public class Links

{

public static void main(String[] args)

{

Node pos1 = null;

Node pos2 = null;

pos1 = new Node(new Integer(13));

pos1.setNext(new Node(new Integer(15), null));

pos2 = new Node(new Integer(11), null);

pos2.setNext(pos1);

printList(pos2);

}

private static void printList(Node head)

{

if (head != null)

{

System.out.println(head.getItem());

printList(head.getNext());

}

}

}

The Links2 Program

public class Links2 {

public static void main(String args[])

{

Node head = null;

head = insert(head, new Integer(13));

head = insert(head, new Integer(-1));

head = insert(head, new Integer(0));

head = insert(head, new Integer(50));

printList(head);

}

private static Node insert(Node head, Comparable newValue)

{

Node prev, curr = head;

for (prev = null, curr = head;

curr != null && newValue.compareTo(curr.getItem()) > 0;

prev = curr, curr = curr.getNext() )

{}

Node newNode = new Node(newValue, curr);

if (prev != null)

{

prev.setNext(newNode);

return head;

}

else

return newNode;

}

private static void printList(Node head)

{

if (head != null)

{

System.out.println(head.getItem());

printList(head.getNext());

}

}

}