Lab Exercises wk12 – Practice with Linked Lists

Required Reading

Chapter 13 - Pointers and Linked Lists

Lecture Slides on Linked Lists, Presented in class wk11

Instructions – PLEASE READ *(notice **bold** and underlined phrases)*

Lab Exercise has three parts:

A. Exercises – Construct a simple Linked List class

B. Submission – Submit your solutions to

1. **You are to work on these lab exercises as individuals.**
2. **Submission deadline: Friday Mar 25th at 10:30am**
3. **The exercises are presented in sequence so that you gradually advance with the material.**
4. **Before you leave the CSIL labs, make sure that a TA looks at your work in order to receive your attendance and lab active participation marks.**
5. **Lab12 Intended learning outcomes**

By the completion of the lab, students should be able to:

- Define a simple linked list
- Write code to display the contents of a linked list
- Add new nodes to the head of the list
- Add new nodes to the end of the list
- Search the list looking for a specific data value
- Insert a new node after a node in the middle of the list

A. Lab Exercises – To be completed by Students Individually

You must complete the exercises and submit your completed linked list functions into CourSys by Friday. You will create a simple Linked List based on the course lecture slides from Wk11.3 slides on Pointers and Linked Lists.

The slides are available in CourSys at:

https://courses.cs.sfu.ca/2016sp-cmpt-135-d1/pages/Wk11.3_PointersLinkedLists.pdf

1. **Download files Node.h, Node.cpp, and Lab12_LinkedListTest.cpp**

   These files are available in CourSys under Lab12. Download them and create a project consisting of these three files.

   `Node.h` defines the `Node` struct and the methods used to add new nodes, search for nodes, and remove nodes. You do not need to modify this file. It already contains the definition for `struct Node` and the function prototypes.

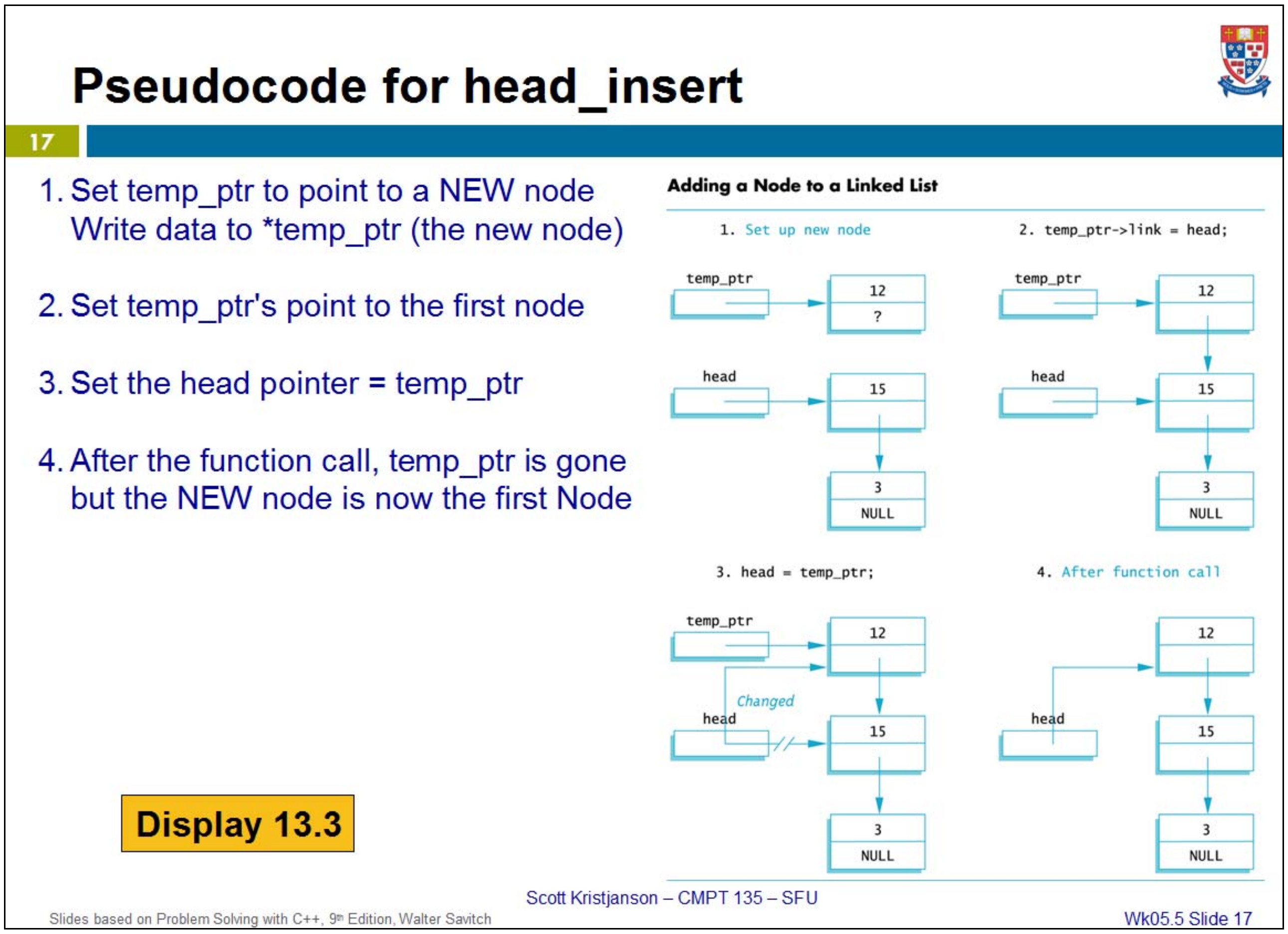
   `Node.cpp` contains stubbed versions of the functions defined in `Node.h`. You must implement these functions in this lab exercise. Having completed this task, you should be ready to attempt the Bonus Question C2 from Assignment 3. You must submit your final working version of `Node.cpp` into CourSys for Lab12.

   The majority of these functions and their implementation are described in the class lecture notes from week 11. Review those slides and translate the pseudo-code presented into actual code within `Node.cpp`.

   `Lab12_LinkedListTest.cpp` is a basic test program for verifying that your functions work as expected. Students are encouraged to enhance this test program to provide more complete testing of your functions. You will not submit this file.

2. Head Insert

Implement the `head_insert` function in `Node.cpp` as described in this slide and slides 16-19. Use the new operator to create a new Node from the heap, update its pointer to point to where head points, then set head to point to this new node.

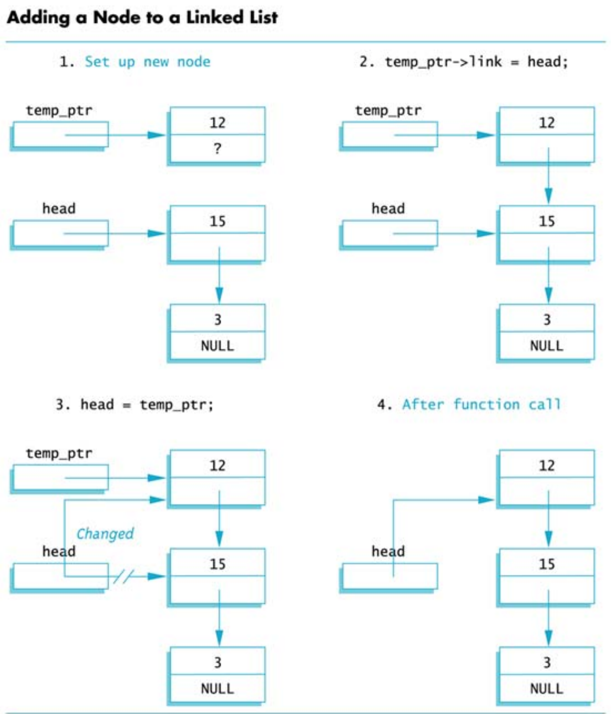


Pseudocode for head_insert

17

1. Set temp_ptr to point to a NEW node
   Write data to *temp_ptr (the new node)
2. Set temp_ptr's point to the first node
3. Set the head pointer = temp_ptr
4. After the function call, temp_ptr is gone
   but the NEW node is now the first Node

Display 13.3



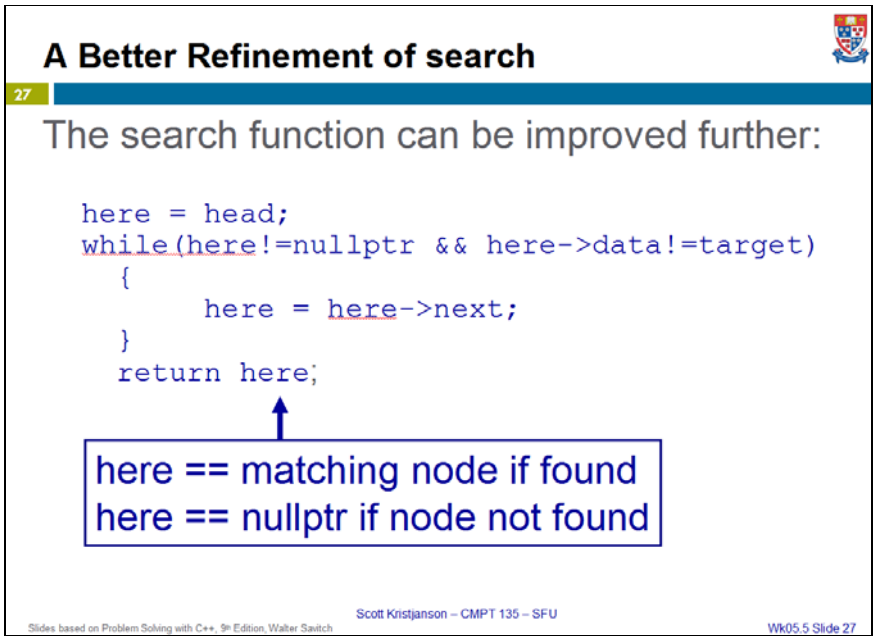
Scott Kristjanson – CMPT 135 – SFU

Slides based on Problem Solving with C++, 9th Edition, Walter Savitch

Wk05.5 Slide 17

3. Search

Implement the `search` function in `Node.cpp` as described in this slide and slides 22-27.



A Better Refinement of search

27

The search function can be improved further:

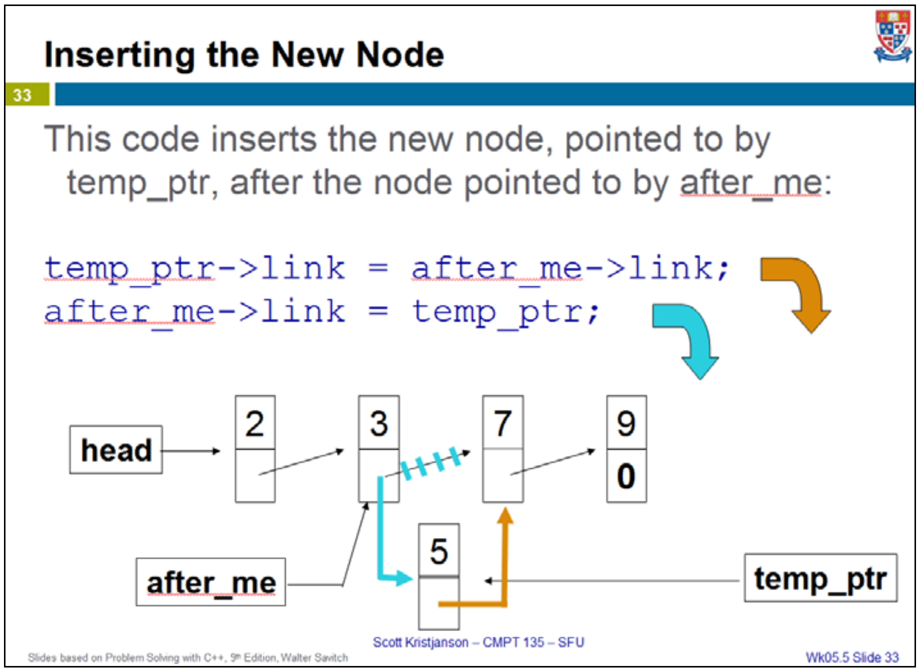
```
here = head;
while(here!=nullptr && here->data!=target)
   {
         here = here->next;
   }
   return here;
```

here == matching node if found
here == nullptr if node not found

Slides based on Problem Solving with C++, 9th Edition, Walter Savitch   Scott Kristjanson – CMPT 135 – SFU   Wk05.5 Slide 27

4. Insert After

Implement the `insert` function as described in slides 31-33.



Inserting the New Node

33

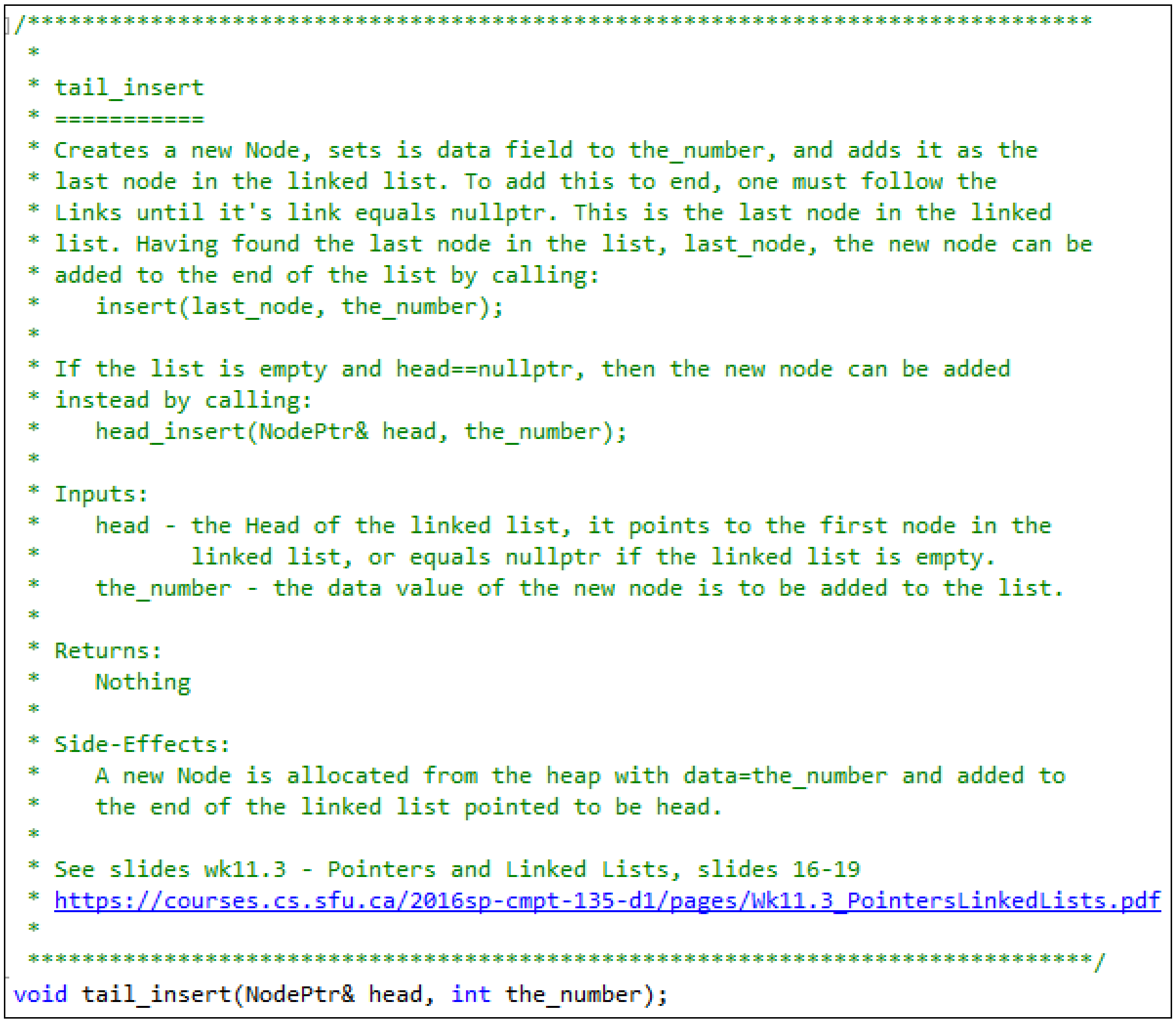
This code inserts the new node, pointed to by temp_ptr, after the node pointed to by after_me:

```
temp_ptr->link = after_me->link;
after_me->link = temp_ptr;
```

Slides based on Problem Solving with C++, 9th Edition, Walter Savitch   Scott Kristjanson – CMPT 135 – SFU   Wk05.5 Slide 33

5. Insert at End of List

Implement the `tail_insert` function as described in `Node.h`. Function `tail_insert` inserts a new Node at the end of a linked list.



```
/*****************************************************************************
 *
 * tail_insert
 * ===========
 * Creates a new Node, sets is data field to the_number, and adds it as the
 * last node in the linked list. To add this to end, one must follow the
 * Links until it's link equals nullptr. This is the last node in the linked
 * list. Having found the last node in the list, last_node, the new node can be
 * added to the end of the list by calling:
 *    insert(last_node, the_number);
 *
 * If the list is empty and head==nullptr, then the new node can be added
 * instead by calling:
 *    head_insert(NodePtr& head, the_number);
 *
 * Inputs:
 *    head - the Head of the linked list, it points to the first node in the
 *           linked list, or equals nullptr if the linked list is empty.
 *    the_number - the data value of the new node is to be added to the list.
 *
 * Returns:
 *    Nothing
 *
 * Side-Effects:
 *    A new Node is allocated from the heap with data=the_number and added to
 *    the end of the linked list pointed to be head.
 *
 * See slides wk11.3 - Pointers and Linked Lists, slides 16-19
 * https://courses.cs.sfu.ca/2016sp-cmpt-135-d1/pages/Wk11.3_PointersLinkedLists.pdf
 *
 *****************************************************************************/
void tail_insert(NodePtr& head, int the_number);
```

6. Removing Nodes

Implement the `removeNode` function as described in `Node.h`. This function searches for a Node with a specific data value and removes it from the linked list.

Implement the `removeHead` function as described in `Node.h`. This function removes the first node from the Linked List and returns the data value stored in that node to the caller.

Implement the `removeTail` function as described in `Node.h`. This function removes the last node from the Linked List and returns the data value stored in that node to the caller.

After removing the Node from the list, these routines call the delete operator to return the memory back to the heap.

7. Output List

Implement the `outputList` function as described in `Node.h`. This function outputs a space separated list of data values starting with the first Node and ending with the last Node in the list. If the Linked List is empty, then the word "empty" is returned.

B. Lab Exercise Submission – To be completed by Students

Students are responsible for submitting the requested work files by the stated deadline for full marks. Since Lab Exercise solutions may be discussed in class following the submission deadline, late submissions will NOT be accepted. It is the student's responsibility to submit on time. If you do not have access to CSIL or issues with the computers in CSIL, please contact the CSIL Help Desk at `helpdesk@cs.sfu.ca`

Students must work on these exercises individually and submit the set of files to CourSys. No group should be created for this submission.

1. **You must submit your final version of the following file before the deadline. Students must ensure that all submitted code compiles and is properly commented and formatted for readability:**
   - `Node.cpp`
2. **Files are to be submitted into CourSys under Lab12.**