Linked Lists

Lecture 04

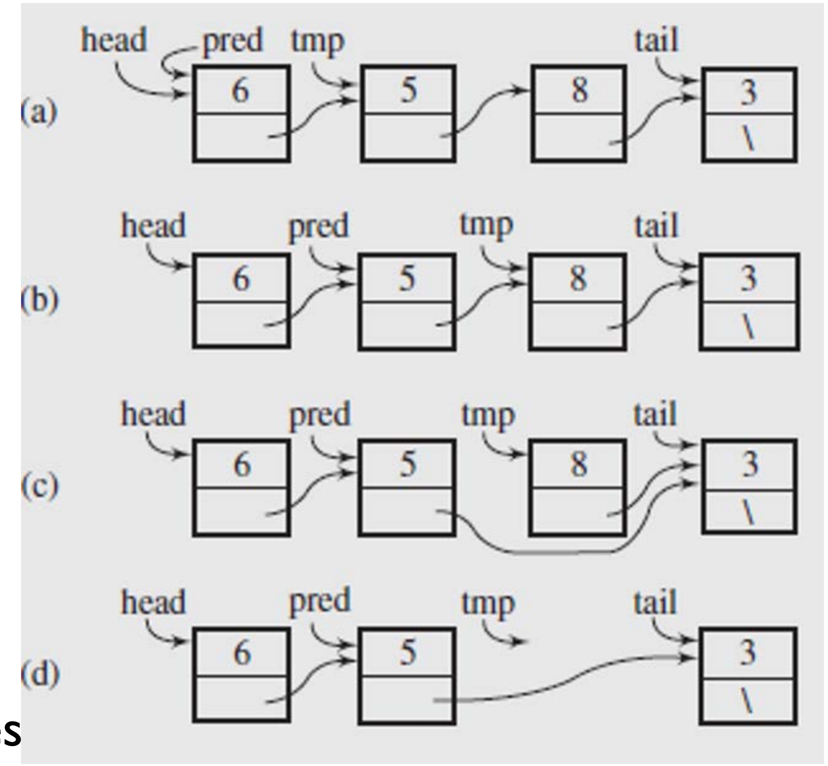
Instructor: Dr. Cong Pu, Ph.D.

cong.pu@okstate.edu

Adapted partially from Data Structures and Algorithms in Java, M.T. Goodrich, R.Tamassia and M. H. Goldwasser, Sixth Edition, Wiley; Data Structures and Algorithms in C++, Adam Drozdek, 4th Edition, Cengage Learning

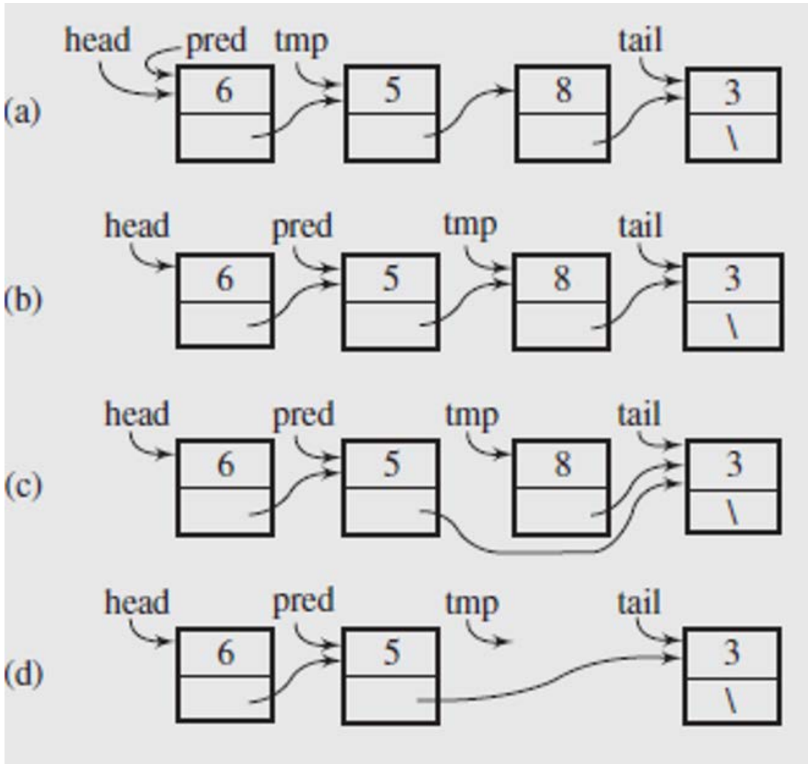
Singly Linked Lists (cont.)

- **Deletion** (cont.): at the middle of a list
  - locate the specific node, then link around it by linking the predecessor of this node directly to its successor
  - need to keep track of the predecessor node, and need to keep track of the node containing the target value
  - require two extra pointers, **pred** and **tmp**, initialized to the first and second nodes in the list, respectively
  - traverse the list until **tmp** → info matches the target value



Singly Linked Lists (cont.)

- **Deletion** (cont.): at the middle of a list
  - set pred → next = tmp → next, "bypasses" the target node, allowing it to be deleted
  - several special cases to consider
    - removing a node from an empty list or trying to delete a value that isn't in the list
    - deleting the only node in the list
    - removing the first or last node from a list with at least two nodes



Singly Linked Lists (cont.)

- **Deletion** (cont.): at the middle of a list

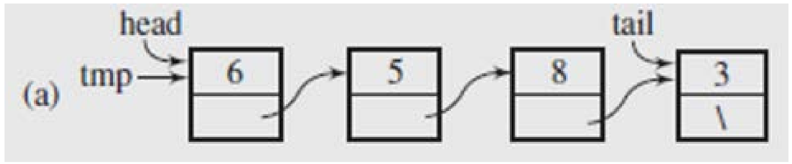
```
void IntSLList::deleteNode(int el) {
    if (head != 0)                         // if non-empty list;
        if (head == tail && el == head->info) { // if only one
            delete head;                          // node on the list;
            head = tail = 0;
        }
        else if (el == head->info) {  // if more than one node on the list
            IntSLLNode *tmp = head;
            head = head->next;
            delete tmp;                // and old head is deleted;
        }
        else {                         // if more than one node in the list
            IntSLLNode *pred, *tmp;
            for (pred = head, tmp = head->next; // and a non-head node
                tmp != 0 && !(tmp->info == el);// is deleted;
                pred = pred->next, tmp = tmp->next);
            if (tmp != 0) {
                pred->next = tmp->next;
                if (tmp == tail)
                   tail = pred;
                delete tmp;
            }
        }
}
```

Singly Linked Lists (cont.)

- **Deletion** (cont.): several special cases for consideration
  - An attempt to remove a node from an empty list, in which case the function is immediately exited
  - Deleting the only node from a one-node linked list
    - both *head* and *tail* are set to null
  - Removing the first node of the list with at least two nodes, which requires updating head
  - Removing the last node of the list with at least two nodes, leading to the update of tail
  - An attempt to delete a node with a number that is not in the list
    - do nothing

Singly Linked Lists (cont.)

- Searching
  - scan a linked list to find a particular data member
  - no modification to the list
    - use a single temporary pointer tmp
    - traverse the list until



      - the info member of the node tmp points to matches the target, or
    - tmp → next is ***null***
      - reached the end of the list and the search fails