CS 162
Intro to Programming II

Linked Lists
Linked Lists

• Is a data structure constructed with pointers
• Implementing a Linked List requires two classes:
  – A **Node** class for each element in a linked list
  – A **List** class for the linked list itself
• Can also be created using structs
Linked Lists

- head is a pointer to the first element in the list
- It is a private member variable of List
- The other boxes are the Nodes, one for each item in the list.
  - Notice data and next (pointer) distinction.
class Node {
  public:
    Node(char dataValue, Node* nextValue) :
      data(dataValue), next(nextValue) {}
    char getData() const { return data; }
    Node* getNext() const { return next; }
    void setData(char theData) { data = theData; }
    void setNext(Node* n) { next = n; }
  private:
    char data;
    Node* next;
};
Linked Lists

class List {
public:
    List() { head = NULL; }
~List();
private:
    Node* head;
};

• Creates and empty List

• empty list is indicated by head = NULL
Linked Lists

• Suppose you call the default constructor:
  ```
  List* l = new List();
  ```
• This creates an empty list
• Pictorially, it looks like:

```
head
NULL
```
Insert a Node

• Insert a Node at the head of an empty list
  1. Create a Node object. Put the data in the data member. Put NULL into the next member.

Remember, you must always have a pointer to any node!
Insert a Node

1. Set head member to point to the new Node.

New is no longer needed.
Insert a Node

- Insert a node at the head of a non-empty list
  1. Create a new Node and fill data member
  2. Set next to node head points to
  3. Set head to point to New
Insert a Node

• The code for both cases is:

```cpp
void List::headInsert(char theData) {
    head = new Node(theData, head);
}
```
Insert a Node

• Insert a node in the middle of a List
  – Suppose you want to insert B between A and C?

Node A is the “afterme” node
1. Create a Node object and put the data into the data member.
2. Set New node next to afterme (A) node next.
3. Set A Node next to New.
Insert a Node

• The code is:

```cpp
void List::insert(Node* afterMe, char theData) {
    afterMe->setNext(new Node(theData,
                              afterMe->getNext()));
}
```
Insert a Node

- The `insert` function works even if the `afterMe` node is the last node in the list.
- The `insert` function cannot insert at the head of the list.
  - Need to call the `headInsert` function.
- How do you get to the `afterMe` node?
  - See the `search` function later on.
Remove a Node

- Removing a Node at the head of the list

1. Assign temp to point to the head Node.
Remove a Node

2. Set head equal to next member of temp.
Remove a Node

3. If needed delete the node pointed to by temp.
Remove a Node

void List::headRemove() throw (Exception){
    if( head != NULL ) {
        Node* temp = head;
        head = head->getNext();
        delete temp;
    } else {
        throw Exception("Can’t remove from empty list");
    }
}

The exception is just one way to test for an empty list before removing a node.
Remove a Node

• Remove a node in the middle of a list
  1. Set before node (A) next equal to discard node (B) next

```
head

A

B

discard

C

NULL
```

19
Remove a Node

2. If needed delete discard.

It is always a good idea to draw these diagrams to check your algorithms, BEFORE writing code.
Remove a Node

```c++
void List::remove(Node* before, Node* discard)
{
    before->setNext(discard->getNext());
    delete discard;
}
```
Searching a List

• How do you find the location of a specified data value in a linked list?
  – Follow the pointers.
  – A linear search algorithm
    • More about this in upcoming lectures
Searching a List

Node* List::search(Node* head, char target) {
    Node* curr = head;
    if( curr == NULL ) {  //Empty list
        return NULL;
    } else {
        while(( curr->getData() != target )
            && ( curr->getNext() != NULL )) {
            curr = curr->getNext();
        }
        if( curr->getData() == target )
            return curr;
        else
            return NULL;
    }
}
Time Complexity

• What is the time complexity of inserting or removing a node into a linked list?
  – Calling search to get to a location in the List:
    \( O(N) \)
  – Actually inserting the node once you have the location:
    \( O(1) \)