Recitation Guide for February 11th, 2008

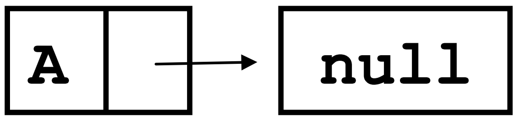
I. Housing Keeping
   a. Bonus Homework 1 – Due tonight.
   b. Homework 3 – Due Friday February 15th.
   c. Quiz 1 – Average: 83 (including zeros)
   d. Exam 1 - Go over solutions.

II. Linked list and homework 3
   a. Linked Lists – a dynamic data structure that stores information through nodes. Each node is a data structure itself that stores information and the next node that it references. Any given linked list is defined by its first node, typically known as the head. This first node has its data element and then points you to the next node. To find the last element in the list you need to start at the first node and traverse the list node by node until you hit the last element whose next node is null.

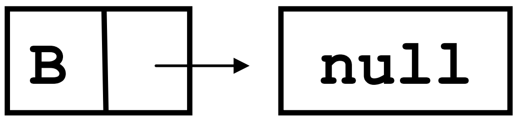
   b. Good real life example: Treasure map

   c. Advantages and disadvantages of Linked Lists
      i. Advantages
         1. dynamic length – can expand to fit more Objects.
         2. Easier to insert in the beginning, middle and the end.
      ii. Disadvantages
         1. dynamic length – length is not as easily obtain compared to an array.
         2. Not easy to index/traverse.
         3. More "complex" conceptually than arrays

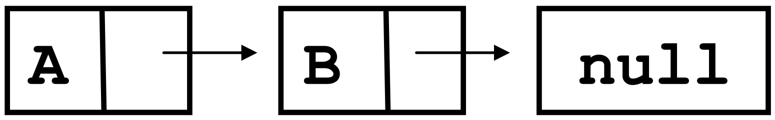
   d. Inserting and deleting in a simple linked list
      i. Create the first node, node1, whose data is A and next is null.



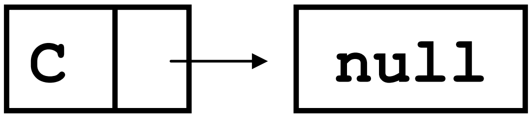
      ii. Create a second node, node2, whose data is B and next is also null.



      iii. To create a simple linked list, have A's next point to B.

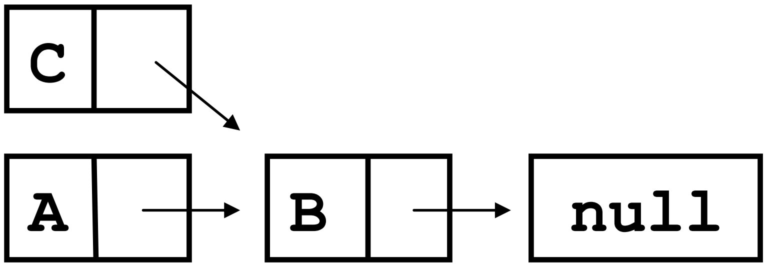


iv. Now suppose we create a third node, node3, whose data is C and next is also null.

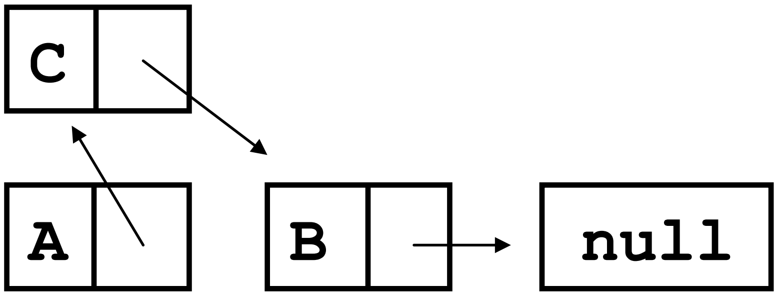


v. Insertion
We want to add node3 after node1.

1. We would first have node3 point to node1's next, which is node2.

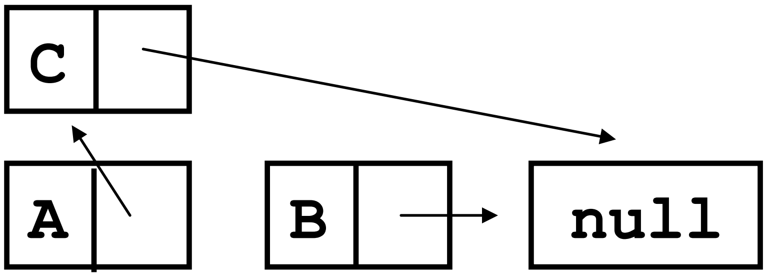


2. Then we would change node1's next to be node3.



vi. Deletion
We want to delete node2.

1. We would first set node3's next to be node2's next, which is null.



2. Then we would set node2 to null but in this case it is already done for us because we removed the last element in the list.

e. Traversing the linked list
    i. Questions to ask yourself before writing your linked list methods:
        1. Do I want to work with the current node or the one after it?
        2. What should my terminating condition be?
        3. What do I want to do at each node?
        4. Do I move to next node after I am done or stay in the same place?
        5. Are there any special cases that might break my code?
            a. Do I notify the user that this case has occurred?
            b. Do I move on after the case has occurred?
    ii. Traversing a listed list a while loop is best because the exact number of times to loop is not always known.
    iii. Generic loops
        1.
```
//This section of code will visit every node
//Initialization step
Node current = head;
//Conditional step
while (current != null){
   //Do something with the current node here
   //Incremental step
   current = current.getNext();
}
```
        Useful if you need to visit every node in the list.

        2.
```
//This section of code will NOT visit every node
/* Do not forget to handle the cases where the head
of the list is the one you want to operate on*/
//Initialization step
Node current = head;
//Conditional step
while (current.getNext() != null){
  //Do something with the current node here
  //Incremental step
  current = current.getNext();
}
```
        Useful if you need to know the node previous to you want to find.
    iv. Just like you have learned in other subjects, there is usually more than one way of doing something. Stop and consider the current situation at hand and answer each of the questions before starting to write your code.
    v. More Resources:
    http://coweb.cc.gatech.edu/cs1316/uploads/336/TraversingTheLinkedList_part1-dsf.2.pdf
    http://coweb.cc.gatech.edu/cs1316/uploads/336/TraversingTheLinkedList_part2-dsf.pdf

f. Accessors and modifiers aka getters and setters
   i. Accessors are methods that access/retrieve private variable values.
      1. Naming convention: get + name of variable
      2. getNext() – return the next node
   ii. Modifiers are methods that set/change private variable values.
      1. Naming convention: set + name of variable
      2. setNext() – sets the next node

g. Useful things to know for the homework
   i. `compareTo` – compares two `Objects` and in general for `String`s, it returns integer value <0 if the former is less than the latter, integer value = 0 if they are equal and integer value > 0 if the former is greater than the latter.
      1. Example:
```
> String s1 = "recitation";
> String s2 = "guide";
> s1.compareTo(s2)
> 11
```
         Because we got an answer >0, this means that `s1` is greater than (later alphabetically) `s2`.
   ii. `equals` – compares for exact equivalence return `true` if they are equal and `false` if they are not.
      1. Example:
```
> s1.equals(s2)
false
> s1.equals(s1)
true
```
   iii. `toString()` – returns a `String` representation of the class
   iv. `System.out.println()` – prints out whatever is taken in as a parameter and then add a carriage return (`\n`).
   v. `System.out.print()` – prints out whatever is taken in as a parameter and does not add a carriage return.
   vi. `\t` – tab character
   vii. `\n` – carriage return character

h. Questions on homework 3?