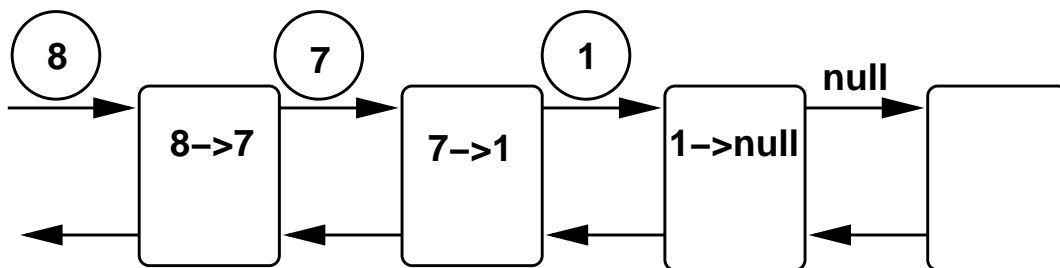
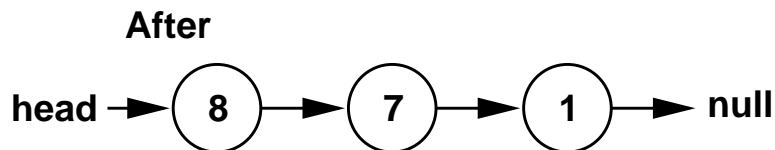
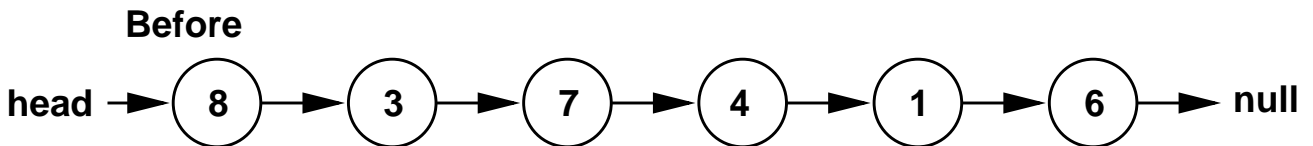


MIDTERM 2 REVIEW

Problem 1. (10 points) You have found the following method in a class called LinkedList. What does this method accomplish if called with `theLL.unknown1(head)`? Don't just translate the code line by line. Describe what it does at a higher level. Under what circumstances will it throw an exception? Explain and show a small example.

```
public void unknown1(Node nd) {  
    if (nd == null)  
        return;  
    nd.setNext(nd.getNext().getNext());  
    unknown1(nd.getNext());  
}
```

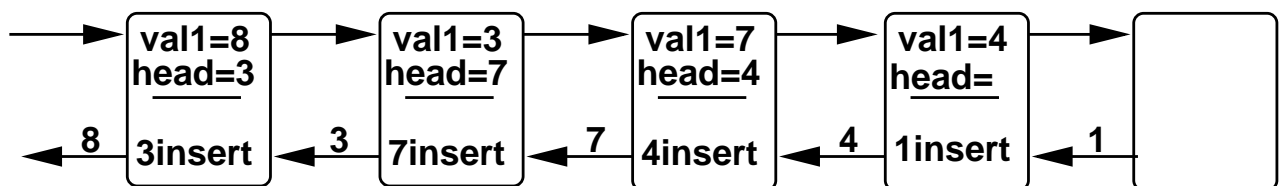
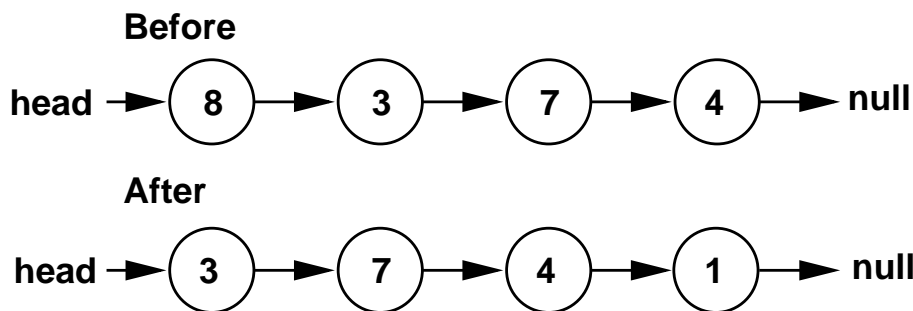
Answer 1. This method deletes every other item in the linked list (if the nodes are number starting with 0 then it deletes the odd numbered nodes). It throws a `NullPointerException` if the number of nodes in the initial linked list is odd.



Problem 2. (10 points) You have found the following method in a class called LinkedList. What happens if the you call **theLL.unknown2()**? Don't just translate the code line by line. Describe what it does at a higher level. Under what circumstances will it throw an exception? Explain and show a small example.

```
public int unknown2() {
    int val1, val2;
    Node nd;
    if (head == null)
        return 1;
    val1 = head.getValue();
    head = head.getNext();
    val2 = unknown2();
    nd = new Node(val2);
    nd.setNext(head);
    head = nd;
    return val1;
}
```

Answer 2. This method deletes the first element in the linked list and inserts 1 at the end. There is no case where it throws an exception.



Problem 3. (10 points) Sort the following array using quick sort (always select the first item as the pivot). Show your intermediate steps.

17	83	41	10	22	67	34	5
----	----	----	----	----	----	----	---

Answer 3.

10	22	5	<u>17</u>	34	67	41	83
----	----	---	-----------	----	----	----	----

5	<u>10</u>	22	<u>17</u>	<u>34</u>	83	41	67
---	-----------	----	-----------	-----------	----	----	----

<u>5</u>	<u>10</u>	<u>22</u>	<u>17</u>	<u>34</u>	41	67	<u>83</u>
----------	-----------	-----------	-----------	-----------	----	----	-----------

<u>5</u>	<u>10</u>	<u>22</u>	<u>17</u>	<u>34</u>	<u>41</u>	67	<u>83</u>
----------	-----------	-----------	-----------	-----------	-----------	----	-----------

<u>5</u>	<u>10</u>	<u>22</u>	<u>17</u>	<u>34</u>	<u>41</u>	<u>67</u>	<u>83</u>
----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------

Problem 4. (10 points) Sort the following array using radix sort. Show your intermediate steps.

10101	00101	11001	01101	11101	11110	00011	11111
-------	-------	-------	-------	-------	-------	-------	-------

Answer 4.

11110	10101	00101	11001	01101	11101	00011	11111
-------	-------	-------	-------	-------	-------	-------	-------

10101	00101	11001	01101	11101	11110	00011	11111
-------	-------	-------	-------	-------	-------	-------	-------

11001	00011	10101	00101	01101	11101	11110	11111
-------	-------	-------	-------	-------	-------	-------	-------

00011	10101	00101	11001	01101	11101	11110	11111
-------	-------	-------	-------	-------	-------	-------	-------

00011	00101	01101	10101	11001	11101	11110	11111
-------	-------	-------	-------	-------	-------	-------	-------

Problem 5. (10 points) Tom and Maxine are both writing a method to sort items. Tom plans to implement insertion sort. Maxine plans to implement merge sort. Under what circumstances should you prefer Tom's plan? Under what circumstances should you prefer Maxine's plan? Explain.

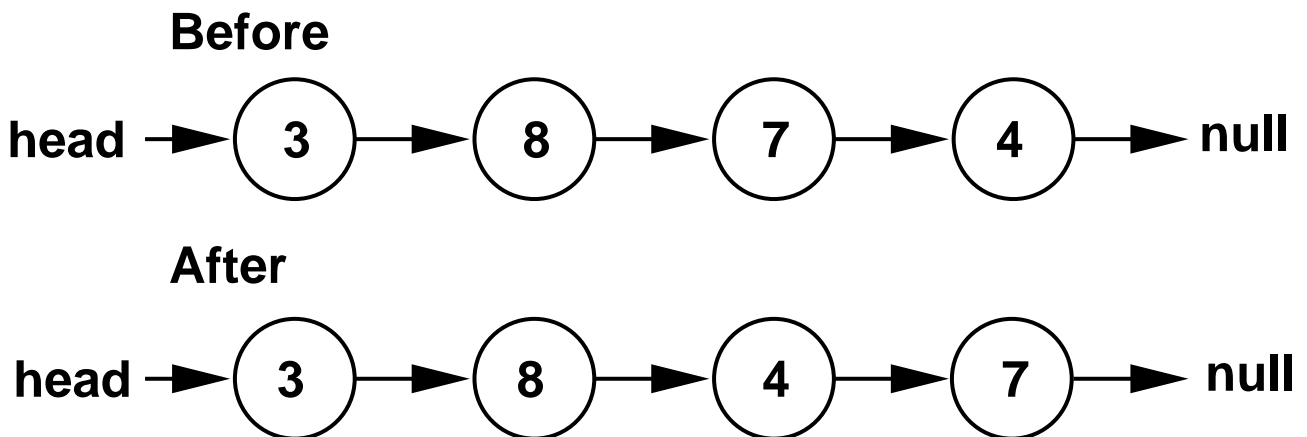
Answer 5. Tom's insertion sort is much simpler to code, but has time complexity $O(n^2)$. Maxine's merge sort is much more complicated, but runs in time $O(n \lg(n))$. Merge sort is almost always implemented using recursion and therefore takes more system resources (stack frames).

If the data set is small or time is not an issue then Tom's is better because it will be faster and easier to code and debug. If the data set is large and speed is crucial then Maxine's is better despite requiring more work to develop and taking up space (stack frames).

Problem 6. (10 points) You have found the following method in a class called Linked List. Describe what would occur if you called `theLL.hmmm()`. Don't just translate the code line by line. Instead describe it at a higher level. Explain and show a small example.

```
public void hmmm(){
    int val1, val2;
    Node nd = head;
    while (nd != null && nd.getNext() != null) {
        val1 = nd.getValue();
        val2 = nd.getNext().getValue();
        if (val1 > val2) {
            nd.setValue(val2);
            nd.getNext().setValue(val1);
        }
        nd = nd.getNext().getNext();
    }
}
```

Answer 6. This method pairs the items (first two, next two, next two, ...) and then swaps the values for those pairs which have the larger one first.



Problem 7. (10 points) Sydney and Vaughn are both writing a program to store information. Sydney plans to use a Linked List. Vaughn plans to use a Hash Table with open addressing. Under what circumstances should you prefer Sydney's plan. Under what circumstances should you prefer Vaughn's plan? Explain.

Answer 7. Sydney's requires understanding references, but can hold arbitrarily many items (limited only by memory), inserting $O(1)$, deleting $O(n)$, finding $O(n)$. Vaughn's doesn't require references, but is a little more sophisticated because of the hash function/collision resolution. Vaughn's size must be fixed before seeing the data and is $O(n)$ in the worst case, but typically $O(1)$ for insert, delete, and find.

If you know nothing about the amount of data and are interested only in worst case performance, Sydney's is better. If you have a general idea of the amount of data and typical behavior is more important, Vaughn's is better.

Problem 8. (10 points) Assume that you have a hash table with hash function $h(\text{key}) = \text{key}^2 \% 11$, a collision resolution scheme of probing (first open addressing method) with $g(n) = 2n$, and table size 11. Start with an empty hash table, perform the following operations, and draw the hash table after every delete operation: insert 4, insert 6, insert 2, delete 6, insert 7, insert 10, insert 11, delete 7, insert 5, insert 8, insert 9, delete 8.

Answer 8.

0	1	2	3	4	5	6	7	8	9	10
			*	2	4					

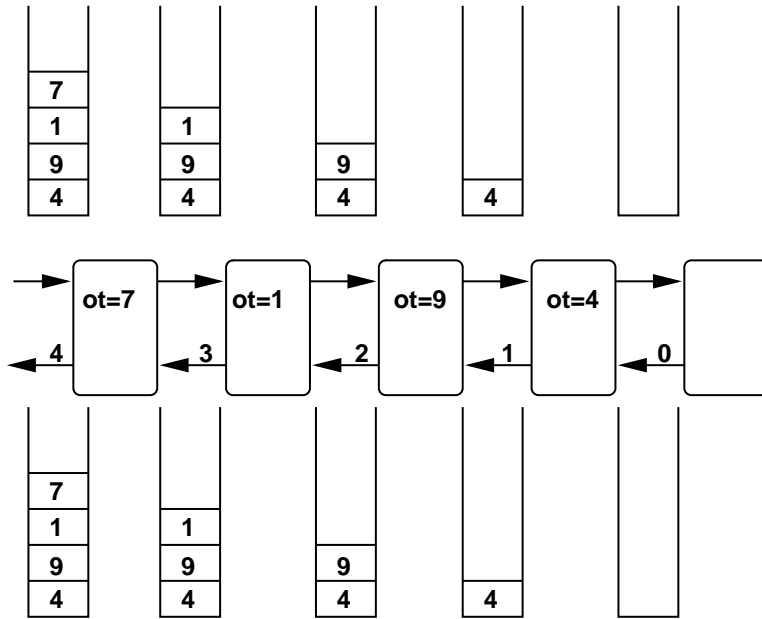
0	1	2	3	4	5	6	7	8	9	10
11	10		*	2	4		*			

0	1	2	3	4	5	6	7	8	9	10
11	10		5	2	4	9	*		*	

Problem 9. (10 points) Assume that the following method is in a class called MyStack. What is the purpose of the method? Do not simply translate the code line by line, instead give a high level description. Show the results of running the method on a small stack.

```
public int hmmm() {
    OurThing ot;
    int n;
    if (isEmpty())
        return 0;
    ot = pop();
    n = hmmm();
    push(ot);
    return n+1;
}
```

Answer 9. This method doesn't change the stack at all, but it returns the number of items in the stack.



Problem 10. (10 points) Assume that you have an empty priority queue (higher values have higher priority). Perform the following operations and show the priority queue after each dequeue. Enqueue(23), Front(), Enqueue(32), Enqueue(3), Front(), Dequeue(), Enqueue(21), Enqueue(52), Enqueue(5), Dequeue(), Enqueue(32), Enqueue(12), Dequeue().

Answer 10.

F 23 3

F 23 21 5 3

F 23 21 12 5 3

Problem 11. (10 points) Let $A = \{5, 3, 6, 7, 2\}$, $B = \{9, 7, 8, 4, 5\}$, and $C = \{9, 7, 2, 8, 1\}$. Find the following 5 sets.

Answer 11.

$$\begin{aligned}
 |A| &= 5 \\
 A \cup C &= \{2, 3, 4, 5, 6, 7, 8, 9\} \\
 B \cap C &= \{7, 8, 9\} \\
 (A \cup B) \cap C &= \{2, 7, 8, 9\} \\
 (A - B) \cup C &= \{1, 2, 3, 6, 7, 8, 9\}
 \end{aligned}$$