Name: Benjamin Lerner
UNI: bll2121

3.1

```
public interface Collection<AnyType> extends Iterable<AnyType> int size( );
boolean isEmpty( ); void clear( );
boolean contains( AnyType x ); boolean add( AnyType x );
boolean remove( AnyType x ); java.util.Iterator<AnyType> iterator( );
 public interface List<AnyType> extends Collection<AnyType> 2{
AnyType get( int idx ); AnyType set( int idx, AnyType newVal );
void add( int idx, AnyType x ); void remove( int idx );
ListIterator<AnyType> listIterator( int pos );
```

```
public class PrintLots implements List {
ArrayList<Integer>L = new ArrayList<Integer>(Arrays.asList(2, 3, 8, 9, 13, 22, 34, 55, 69, 80, 90,
100));
ArrayList<Integer>P = new ArrayList<Integer>(Arrays.asList(1,3,4,6));
for (int i = 0; i < P.size(); ++i) {
        System.out.println(L.get(P.get(i));
}
```

The running time of this procedure would be O(N), where N is P.size(). Every time L.get(P.get(i)), two operations are performed. This is performed for every value in P, giving a time of O(2N), or O(N).

3.2

a)

SingleLinkedList = {node1, node2, node3, node4}
node1.next = node2, node2.next = node3, node1.next.next = node3, etc.

```
public nodeSwap() {
node1.next = node1.next.next;
node2.next = node2.next.next;
node3.next = node1.next;

}
```

output SingleLinkedList = {node1, node3, node2, node4}

b)

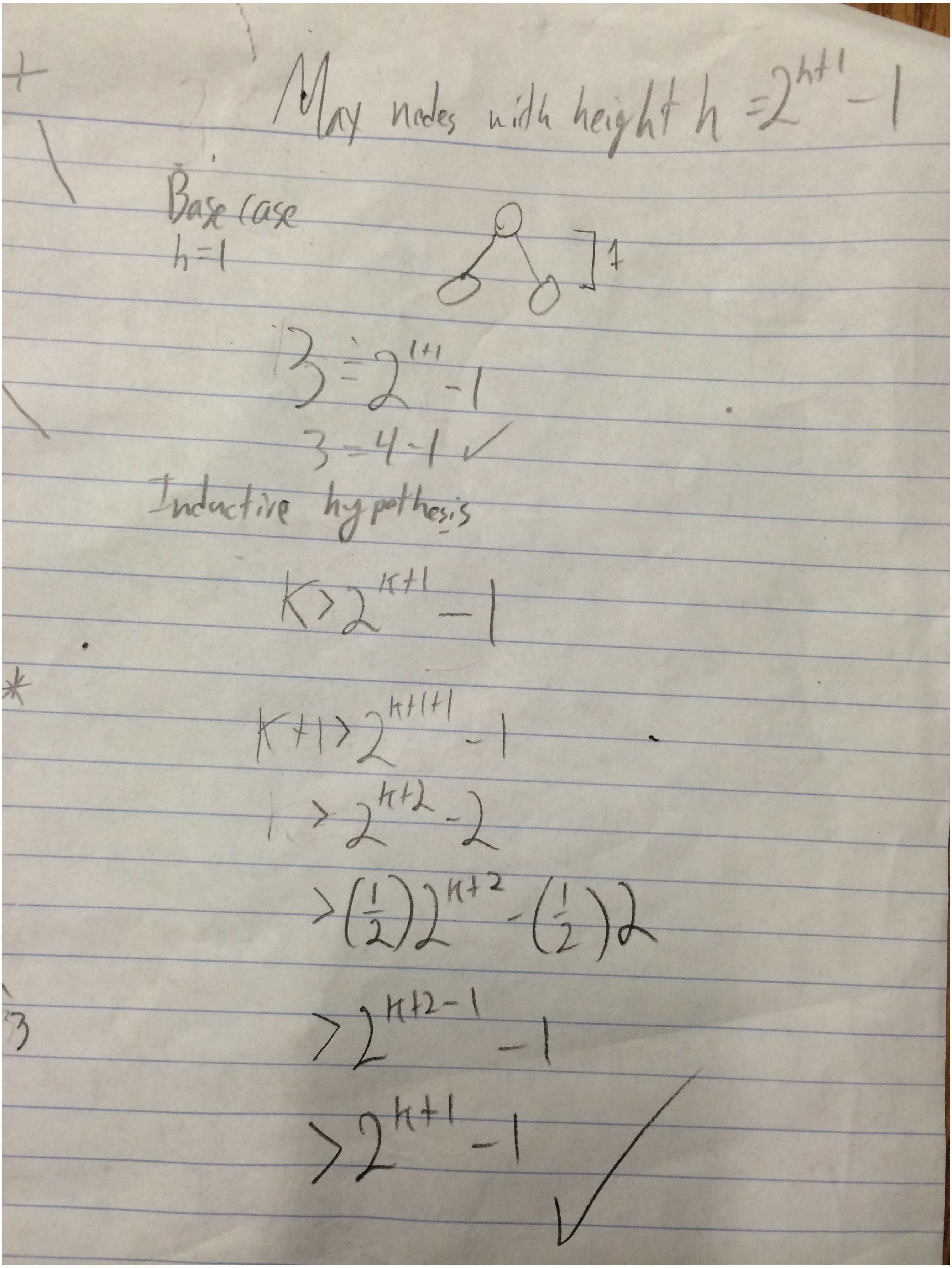
DoubleLinkedList = {node1, node2, node3, node4}
node2.next = node3, node2.prev = node1, node2.next.next = node4, node3.prev.prev = node1, etc.

```
public nodeSwap(node2, node3) {
firstTemp = node2.prev;

node2.next = node3.next
node2.prev = node3;

node3.next = node2;
node3.prev = firstTemp;

node1.next = node3;
node4.prev = node2;
}
```

3.24

```
Array[] arr = {1, 2, 3, … N}
Stack bottomStack, topStack
if (arr. length % 2 == 0) {
for (int i = 0, i < arr.length; ++i) {
        bottomStack.add(arr[i])
        topStack.add(arr[N - i])
}
else {
        for(int i = 0, i < arr.length / 2; ++i) {
        bottomStack.add(arr[i])
}
        for(int j = 0, j < arr.length / 2 + 1; ++j) {
        topStack.add(arr[N - 1])
}
}
```

4.5



Max nodes with height $h = 2^{h+1} - 1$

Base case
$h = 1$

$3 = 2^{1+1} - 1$

$3 = 4 - 1$ ✓

Inductive hypothesis

$k > 2^{k+1} - 1$

*

$k + 1 > 2^{k+1+1} - 1$

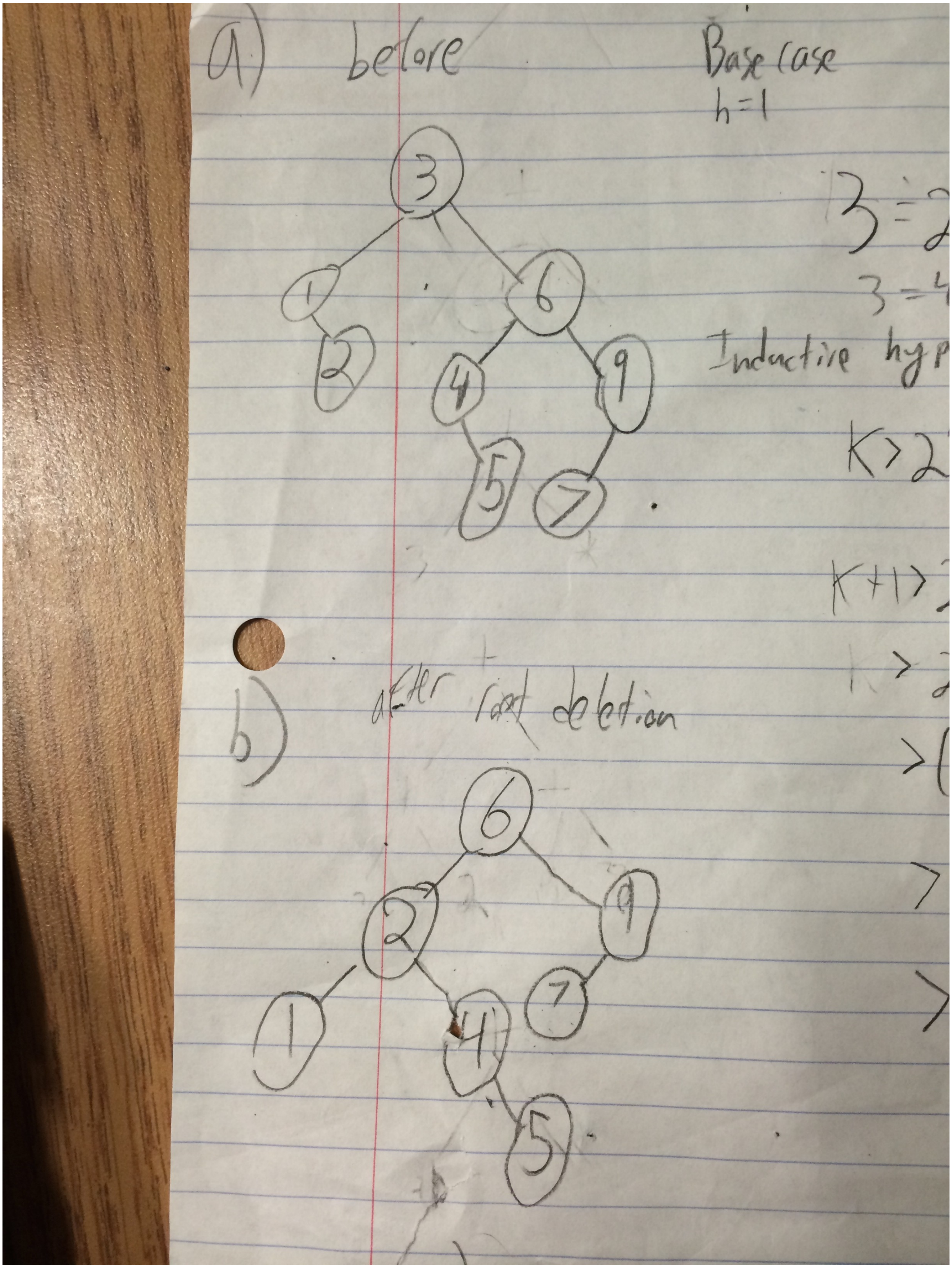
$k > 2^{k+2} - 2$

$> \left(\frac{1}{2}\right)2^{k+2} - \left(\frac{1}{2}\right)2$

$> 2^{k+2-1} - 1$

$> 2^{k+1} - 1$ ✓

4.9



a) before

Base case
h=1

$3 = 2$

$3 = 4$

Inductive hyp

$K > 2$

$K+1 >$

$> 2$

$> ($

$>$

$>$

b) after root deletion