CS307 Fall 2010 Final Solution and Grading Criteria.

Grading acronyms

ABA - Answer by Accident
AIOBE - Array Index out of Bounds Exception may occur
BOD - Benefit of the Doubt. Not certain code works, but, can't prove otherwise
ECF - Error carried forward.
Gacky or Gack - Code very hard to understand even though it works or solution is not elegant. (Generally no points off for this.)
GCE - Gross Conceptual Error. Did not answer the question asked or showed fundamental misunderstanding
NAP - No answer provided. No answer given on test
NN - Not necessary. Code is unneeded. Generally no points off
NPE - Null Pointer Exception may occur
OBOE - Off by one error. Calculation is off by one.

1. As written or -1.5. No partial credit unless stated. On Big O, missing O() is okay.

```
A.                  97
                   /
                 35
                /  \
            -59     93
               \      \
                2      95
```

```
B. 3
C. 50 -7 -50 25 70 90 91
D. -50 -7 25 50 70 90 91
E. -50 25 -7 91 90 70 50
F. Yes
G. No, path rule violated or different nums of black nodes in paths
H. 20007
I. Any answer between 14 and 30 inclusive
J. Ensure BST, no partial credit
K. 3
L. O(N)
M. O(N)
N. O(NlogN) base 2 okay
O. O(NlogN) base 2 okay
P. 18
Q. wwiwrwtwhw
R. Linked list
S. Polymorphism. variables of type Object can refer to any type of
object in java. (Or words to that affect.)
T. 1235
```

2. Suggested solution:

```java
public boolean removeLast(E obj){
    boolean found = false;
    DoubleListNode<E> temp = header.getPrev();
    while(!found && temp != header) {
        if(obj.equals(temp.getValue())) {
            found = true;
            temp.getPrev().setNext(temp.getNext());
            temp.getNext().setPrev(temp.getPrev());
        }
        else
            temp = temp.getPrev();
    }
    return found;
}
```

Criteria. 15 points
start at end of list: 1
use temp to move through list: 2, 3
stop when reach head or found: 3
use equals / check if found: 1
update links correctly if found: 4
return correct value: 1

3A. Suggested Solution:

```
// pre: size != 0, data is in the tree
private void fixNumDescendants(E data) {
    BSTNode<E> temp = root;
    while(!temp.getValue().equals(data)) {
        temp.setNumDescendants(temp.getNumDescendants() + 1);
        int dir = data.compareTo(temp.getValue());
        if(dir < 0)
            temp = temp.getLeft();
        else
            temp = temp.getRight();
    }
}
```

Criteria. 10 points
use temp and start at root: 2
check correctly to stop (== data okay, if null, must not increment last node), 2
increment numDescendants: 1
check direction to go: 2
move correct direction: 3

all nodes checked: -4
alter tree structure: -3
recursive okay if works

3B. Suggested Solution.

```
private E getKthHelp(BSTNode<E> n, int k) {
    int leftCount = 0;
    if(n.getLeft() != null)
        leftCount = n.getLeft().getNumDescendants();
    if(k - leftCount == 1)
        return n.getValue();
    else if(k <= leftCount)
        return getKthHelp(n.getLeft(), k);
    else
        return getKthHelp(n.getRight(),k - leftCount - 1);
}
```

Criteria. 15 points

Check if left exists: 2
get left count: 2
calc if current node is correct based on left count: (base case). 5
if k less than left count go left. k unchanged: 3
if k greater than left count + 1 go right, k = k - leftCount - 1: 3

checks if null okay, but not needed.

O(N) -7

4. Suggested Solution

```
public static void radixSort(int[] data){
    ArrayData<Queue<Integer>> queues
        = new ArrayData<Queue<Integer>>();
    for(int i = 0; i < 10; i++)
        queues.add( new Queue<Integer>() );
    int max = -1; // all elements in data >= 0
    for(int x : data)
       if(x > max)
           max = x;
    int passes = (max + "").length();
    for(int i = 1; i <= passes; i++){
        for(int x : data)
            queues.get(getDigit(x,  i)).enqueue(x);
        int pos = 0;
        for(Queue<Integer> q : queues)
            while( !q.isEmpty())
               data[pos++] = q.dequeue();
    }
}
```

Criteria:
Create ArrayList of Queues: 1
Add 10 new Queues to ArrayList: 3
Find max value and num passes: 4
outer loop for number of passes: 3
first inner loop for all values in array: 2, getDigits: 2: enqueued to correct queue: 3
second inner loop for each queue: 2
for each queue empty into array at correct position: 5

5. Suggested Solution:

```
public static int frequency(Stack<String> st,
                                    String target) {
    Stack<String> temp = new Stack<String>();
    int result = 0;
    while(!st.isEmpty()){
        temp.push(st.pop());
        if(temp.top().equals(target))
            result++;
    }
    while(!temp.isEmpty())
        st.push(temp.pop());
    return result;
}
```

Criteria. 10 points

create temp stack: 1
empty st into temp: 3
check elements equal to target and increment counter: 2 (-1 if ==)
empty tmep back into st correctly: 4
return correct result: 1