Name: Key

Email address: _____

CSE 373 Spring 2009: Midterm #1

(closed book, closed notes, NO calculators allowed)

Instructions: Read the directions for each question carefully before answering. We may give partial credit based on the work you **write down**, so if time permits, show your work! Use only the data structures and algorithms we have discussed in class or which were mentioned in the book so far.

Note: For questions where you are drawing pictures, please circle your final answer for any credit.

Good Luck!

Total: 78 points. Time: 50 minutes.

Question	Max Points	Score
1	16	
2	8	
3	16	
4	8	
5	12	
6	18	
Total	78	

1. (16 pts) **Big-O**

For each of the functions *f(N)* given below, indicate the tightest bound possible (in other words, giving $O(2^N)$ as the answer to every question is not likely to result in many points). Unless otherwise specified, all logs are base 2. You MUST choose your answer from the following (not given in any particular order), each of which could be re-used (could be the answer for more than one of a) – h)):

$O(N^2)$, $O(N^3 \log N)$, $O(N \log N)$, $O(N)$, $O(N^2 \log N)$, $O(N^5)$, $O(2^N)$, $O(N^3)$, $O(\log N)$, $O(1)$, $O(N^4)$, $O(N^N)$, $O(N^6)$, $O(N^8)$, $O(N^9)$

You do not need to explain your answer.

a) *f(N)* = $(N + N + N + N)^2$ — $O(N^2)$

$(4N)^2$

b) *f(N)* = $(N^2 + N) / N$ — $O(N)$

$N + 1$

c) *f(N)* = $(N/2) \log (2N) + N$ — $O(N \log N)$

d) *f(N)* = $N \log N + N \log (N^2)$ — $O(N \log N)$

e) *f(N)* = $(N/3) \log (N^3) + 3N$ — $O(N \log N)$

f) *f(N)* = $N \log N + 1000^4 + N^2$ — $O(N^2)$

g) *f(N)* = $(N \cdot N \cdot N)^2$ — $O(N^6)$

$(N^3)^2$

h) *f(N)* = $256 \log_{256} (N)$ — $O(\log N)$

2. **(8 pts) Big-Oh and Run Time Analysis:** Describe the worst case running time of the following pseudocode functions in Big-Oh notation in terms of the variable n. ***Showing your work is not required*** (although showing work ***may*** allow some partial credit in the case your answer is wrong – don't spend a lot of time showing your work.). You MUST choose your answer from the following (not given in any particular order), each of which could be re-used (could be the answer for more than one of I. – IV.):

   $O(n^2)$, $O(n^3 \log n)$, $O(n \log n)$, $O(n)$, $O(n^2 \log n)$, $O(n^5)$, $O(2^n)$, $O(n^3)$, $O(\log n)$, $O(1)$, $O(n^4)$, $O(n^n)$

I.
```
void silly(int n) {
   for (int i = 0; i < 1000; ++i) {
      for (int j = 0; j < n; ++j) {
         for (int k = 0; k < j; ++k)
            System.out.println("k = " + k);
         for (int m = 0; m < i; ++m)
            System.out.println("m = " + m);
      }
   }
}
```
Runtime: $O(N^2)$

II.
```
void silly(int n, int m) {
   if (n < 0) return;
   if (n < m)
     silly(n/2, m+1);
   else
     silly(n/2, m);
}
```
$O(\log N)$

III.
```
void silly(int n, int x, int y) {
   for (int i = 0; i < n; ++i) {
      if (x < y) {
         for (int j = 0; j < n * n; ++j)
            System.out.println("j = " + j);
      } else
         System.out.println("i = " + i);
   }
}
```
$O(N^3)$

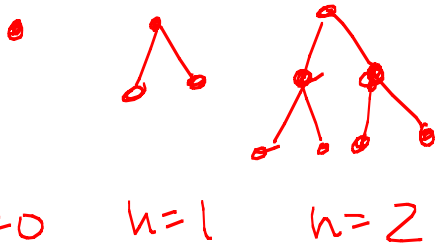
IV.
```
void silly(int n) {
   for (int i = 0; i < n; ++i) {
      j = 0;
      while (j < n) {
         System.out.println("j = " + j);
         j++;
      }
   }
}
```
$O(N^2)$

3. (16 pts) **Trees**.

a) (3 pts) What is the minimum and maximum number of nodes in a **perfect** binary tree of height h (use h in your answer)? (Note: You must solve any summations or recurrences for full credit.)

Minimum = $2^{h+1} - 1$

Maximum = $2^{h+1} - 1$



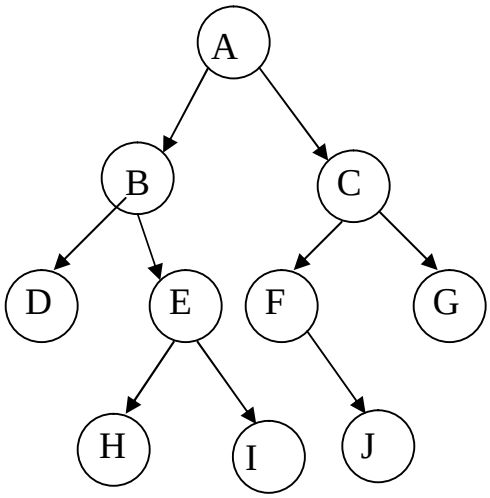
b) (6 pts) Give traversals of the tree shown at the bottom of this page:

Post-Order: D, H, I, E, B, J, F, G, C, A

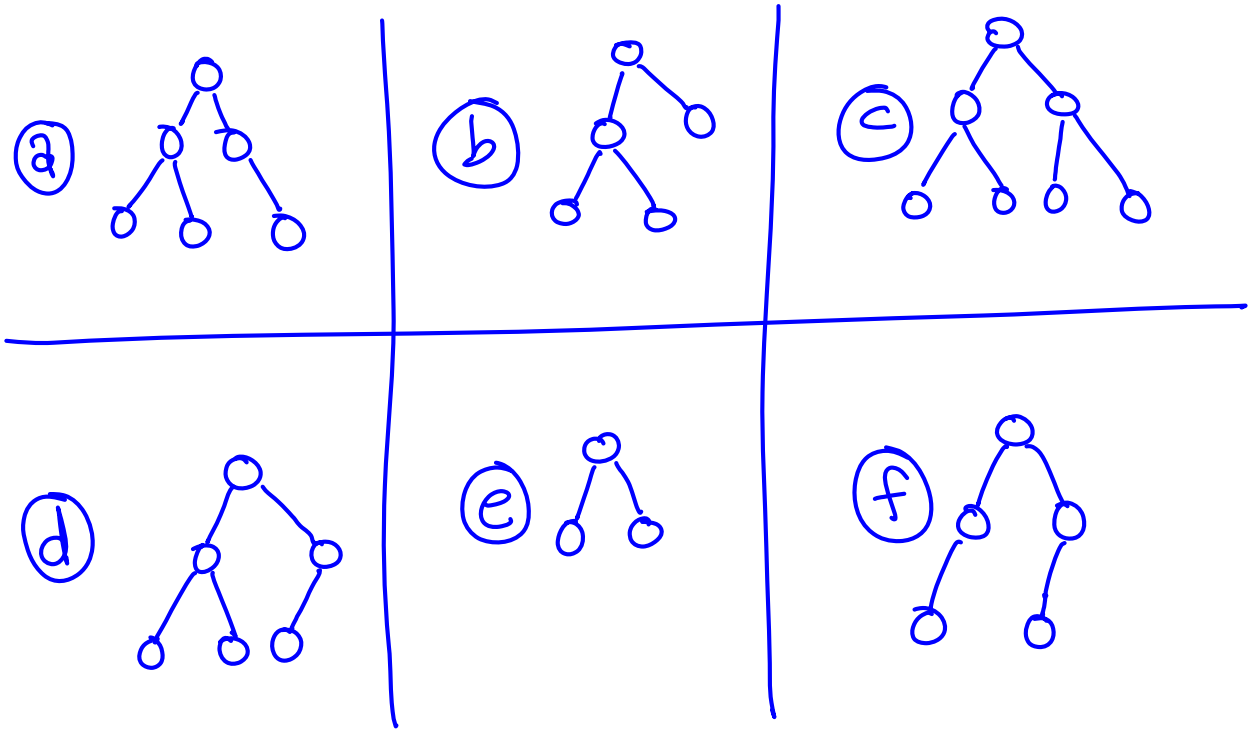
Pre-Order: A, B, D, E, H, I, C, F, J, G

In-Order: D, B, H, E, I, A, F, J, C, G

c) (1 pts) What is the height of the tree shown below: 3



d) (6 pts) Given the following six trees a through f:



List the letters of all of the trees that have the following properties: (Note: It is possible that none of the trees above have the given property, it is also possible that some trees have more than one of the following properties.)

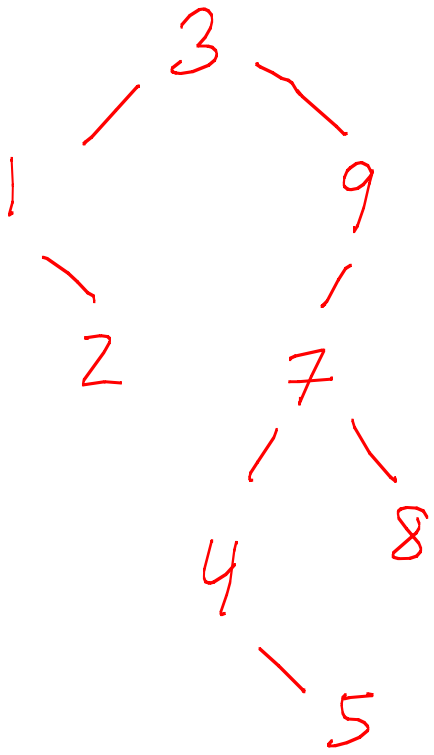
Full: b, c, e

Complete: b, c, d, e

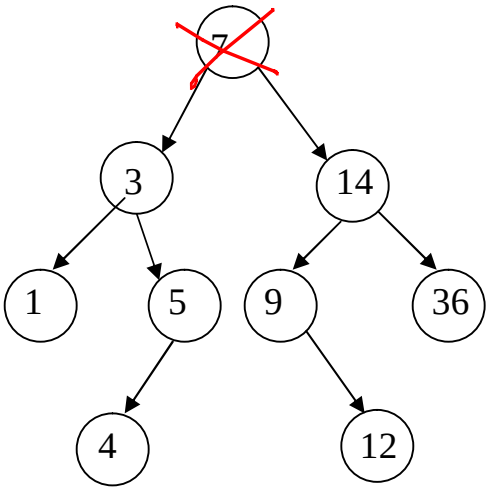
Perfect: c, e

4. (8 pts total) **Binary Search Trees**

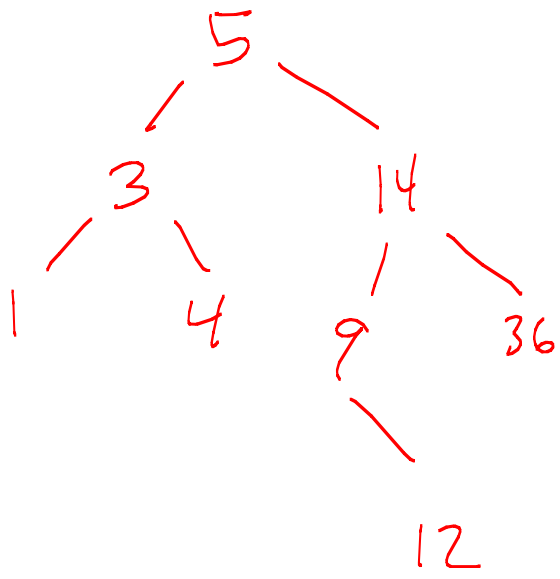
a) (4 pts) **Draw the Binary Search tree** that results from inserting the keys:
3, 9, 1, 7, 4, 5, 8, 2 in that order into an initially empty tree. You are only required to show the final tree, although drawing intermediate trees may result in partial credit. If you draw intermediate trees, ***please circle your final tree for ANY credit.***



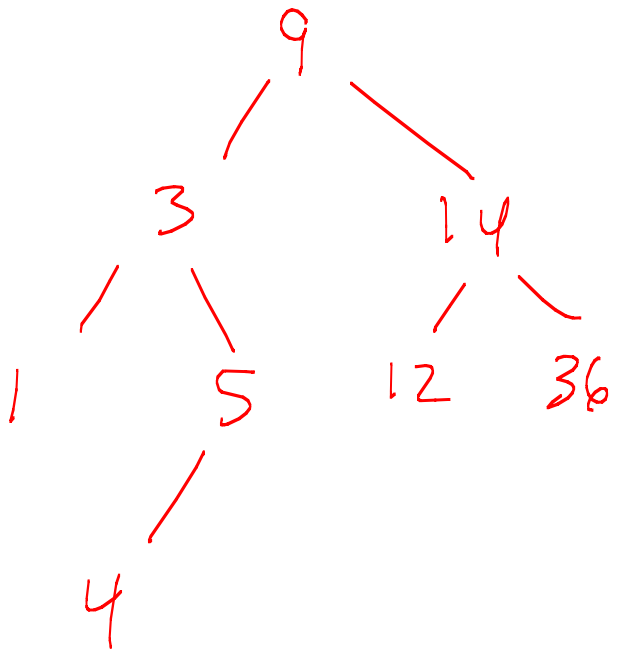
b) (4 pts) Given the tree shown below. Draw what the tree would look like after deleting the value 7. Use one of the methods for deleting described in class or in the book.



Replace 7 with 5



Replace 7 with 9



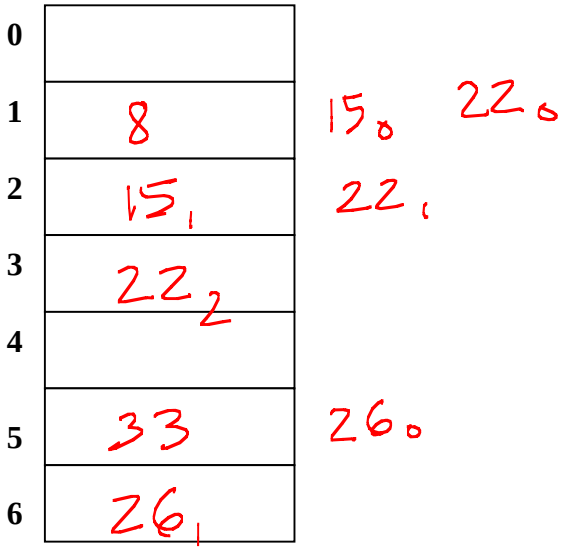
5. (12 points total) **Hashing**:

a) (5 pts) Draw the contents of the hash table in the boxes below given the following conditions:

The size of the hash table is 7.
Open addressing and **linear probing** is used to resolve collisions.
The hash function used is H(k) = k mod 7

What values will be in the hash table after the following sequence of insertions? Draw the values below, and show your work for partial credit.

8, 33, 15, 26, 22



b) (2 pts) What is the **load factor** for the table above?

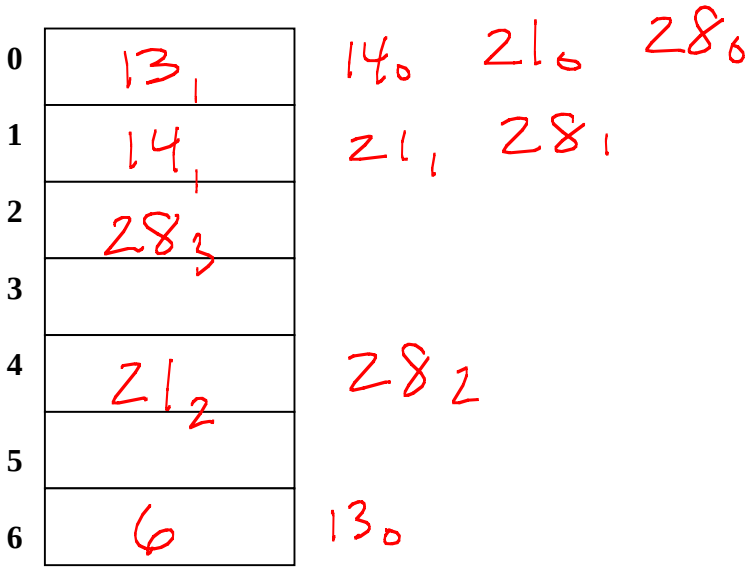
5/7

c) (5 pts) Draw the contents of the hash table in the boxes below given the following conditions:

The size of the hash table is 7.
Open addressing and **quadratic probing** is used to resolve collisions.
The hash function used is H(k) = k mod 7

What values will be in the hash table after the following sequence of insertions? Draw the values below, and show your work for partial credit.

6, 13, 14, 21, 28



6. (18 pts) **Running Time Analysis:** Give the tightest possible upper bound for the ***worst case*** running time for each of the following in terms of *N*. You MUST choose your answer from the following (not given in any particular order), each of which could be re-used (could be the answer for more than one of a) – f)):

$O(N^2)$, $O(N^3 \log N)$, $O(N \log N)$, $O(N)$, $O(N^2 \log N)$, $O(N^5)$, $O(2^N)$, $O(N^3)$, $O(\log N)$, $O(1)$, $O(N^4)$, $O(N^N)$, $O(N^6)$

****For any credit, you must explain your answer.** *Assume that the most time-efficient implementation is used.* Assume that all keys are distinct.

a) Dequeue from a queue containing N elements implemented as a singly-linked list.
Explanation:

Dequeue removes the item from front of queue. Only requires moving a pointer & returning the value – constant time operations.

a) $O(1)$

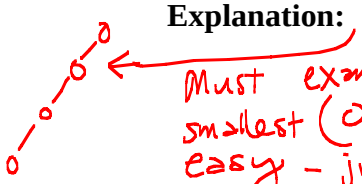
b) Printing out the values stored in all the *leaves* of a binary search tree containing N elements. **Explanation:** In all cases, must visit all interior nodes to find all leaf nodes. This can be implemented w. any traversal plus check for leaf (both L&R ptrs == NULL). Time is same as for a traversal/visiting each node = $O(N)$

b) $O(N)$

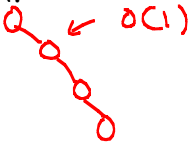
c) Figuring out what the *minimum value* is in a hash table of tablesize N, containing N elements where separate chaining is used and each bucket points to an unsorted linked list. **Explanation:** Must search all buckets ($O(N)$) as well as search all values in all buckets ($O(N)$ values total) until we know we have found smallest value.
$O(N) + O(N) = O(N)$

c) $O(N)$

d) Finding and deleting the smallest value in a binary search tree of size N.
Explanation: Worst case behavior: Must examine all values to find smallest ($O(N)$). Deleting is then easy – just remove element ($O(1)$) $O(N) + O(1) = O(N)$



Note:



d) $O(N)$

e) MakeEmpty on a stack currently containing N/2 elements, implemented as an array of size N. **Explanation:** All that is needed is setting top = 0 (the bottom index in stack). No need to zero out values or to create a new array. Constant time operation to reset top. (or to pop all values off stack)

e) $O(1)$

f) Inserting a value into a perfect binary search tree of size N.
Explanation: A perfect tree has height $O(\log N)$. Any new value inserted must be a child of a leaf. So $O(\log N)$ to find the right location (leaf node) to insert value under, constant time to create a new node & put value in.

f) $O(\log N)$