Name: _____Key_____

Email address: _____

CSE 373 Autumn 2010: 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: 75 points. Time: 50 minutes.

Question	Max Points	Score
1	16	
2	8	
3	15	
4	4	
5	6	
6	8	
7	18	
Total	75	

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)):

$O(N^2)$, $O(N^{1/2})$ $O(N^3 \log N)$, $O(N \log N)$, $O(N)$, $O(N^2 \log N)$, $O(N^5)$, $O(2^N)$, $O(N^3)$, $O(\log N)$, $O(1)$, $O(N^4)$, $O(N^{12})$ $O(N^N)$, $O(N^6)$, $O(N^8)$, $O(N^9)$, $O(N^{10})$

You do not need to explain your answer.

a) $f(N) = (1/2) (N \log N) + (\log N)^2$
$O(N \log N)$

b) $f(N) = N^2 \cdot (N + N \log N + 1000)$
$O(N^3 \log N)$

c) $f(N) = N^2 \log N + 2^N$
$O(2^N)$

d) $f(N) = ((1/2) (3N + 5 + N))^4$
$O(N^4)$

e) $f(N) = (2N + 5 + N^4) / N$
$O(N^3)$

f) $f(N) = \log_{10}(2^N)$
$O(N)$

g) $f(N) = N! + 2^N$
$O(N^N)$

h) $f(N) = (N \cdot N \cdot N \cdot N + 2N)^2$
$O(N^8)$

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 < n; ++i) {
        j = n;
        while (j > 0) {
            System.out.println("j = " + j);
            j = j - 2;
        }
    }
}
```
Runtime:

$O(n^2)$

II.
```
void silly(int n, int x, int y) {
    for (int k = n; k > 0; k--)
        if (x < y + n) {
            for (int i = 0; i < n; ++i)
                for (int j = 0; j < i; ++j)
                    System.out.println("y = " + y);
        } else {
            System.out.println("x = " + x);
        }
}
```

$O(n^3)$

III.
```
void silly(int n) {
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < n; ++j)
            System.out.println("j = " + j);
        for (int k = 0; k < i; ++k) {
            System.out.println("k = " + k);
            for (int m = 0; m < 100; ++m)
                System.out.println("m = " + m);
        }
    }
}
```

$O(n^2)$

IV.
```
int silly(int n, int m) {
    if (m < 2) return m;
    if (n < 1) return n;
    else if (n < 10)
        return silly(n/m, m);
    else
        return silly(n - 1, m);
}
```

$O(n)$

3. (15 pts total) **Trees**.

a) (4 pts) What is the minimum and maximum number of nodes in an **AVL tree of height 6?** (Hint: the height of a tree consisting of a single node is 0) *Give an exact number* for both of your answers – not a formula.

Minimum = 33

$(S(h) = S(h-1) + S(h-2) + 1)$

Maximum = 127

$(2^{n+1} - 1 = 2^7 - 1 = 128 - 1)$

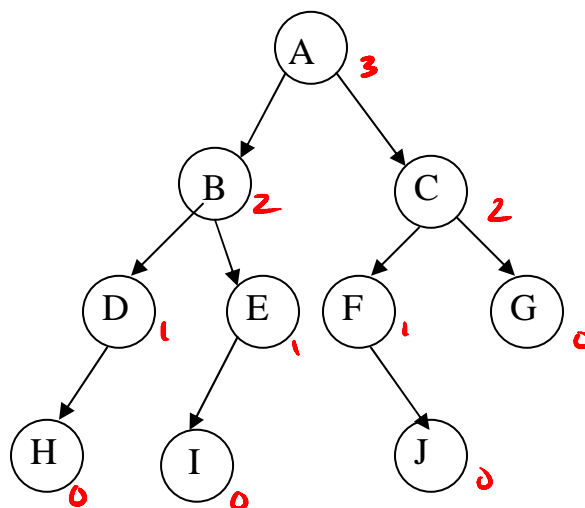b) (3 pts) Give traversals of the tree shown at the bottom of this page:

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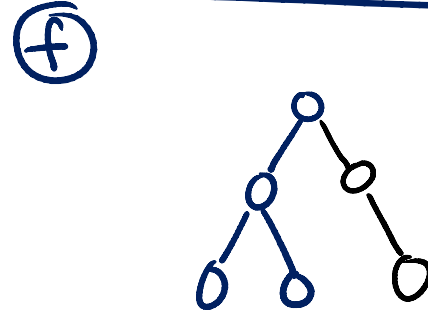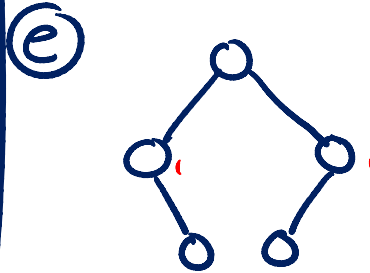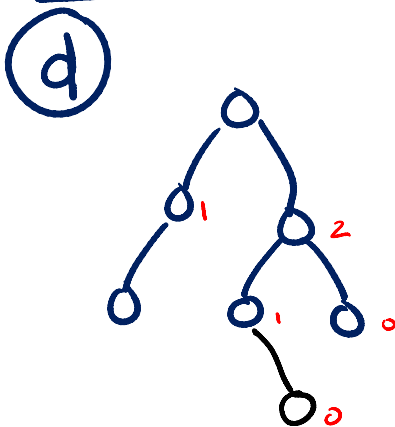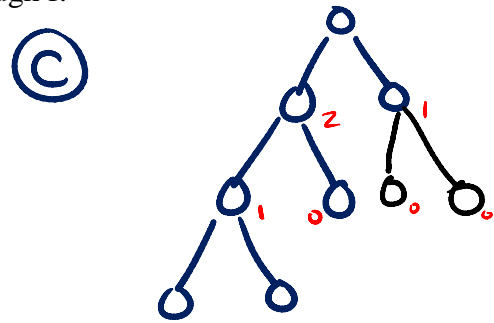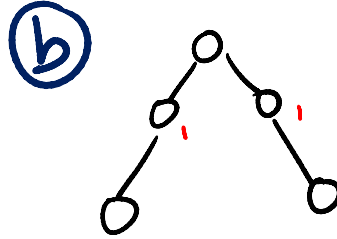
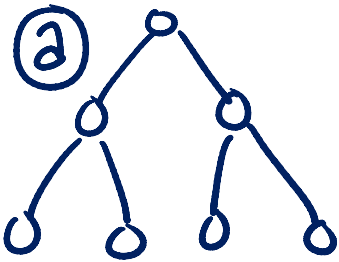Post-Order: H D I E B J F G C A

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

c) (1 pt) What is the **depth** of node F in the tree shown below: 2

d) (1 pt) Is it AVL balanced (ignore the values, only look at the shape):

YES / NO

A 3

B 2   C 2

D 1   E 1   F 1   G 0

H 0   I 0   J 0

3. (cont) e) (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.)
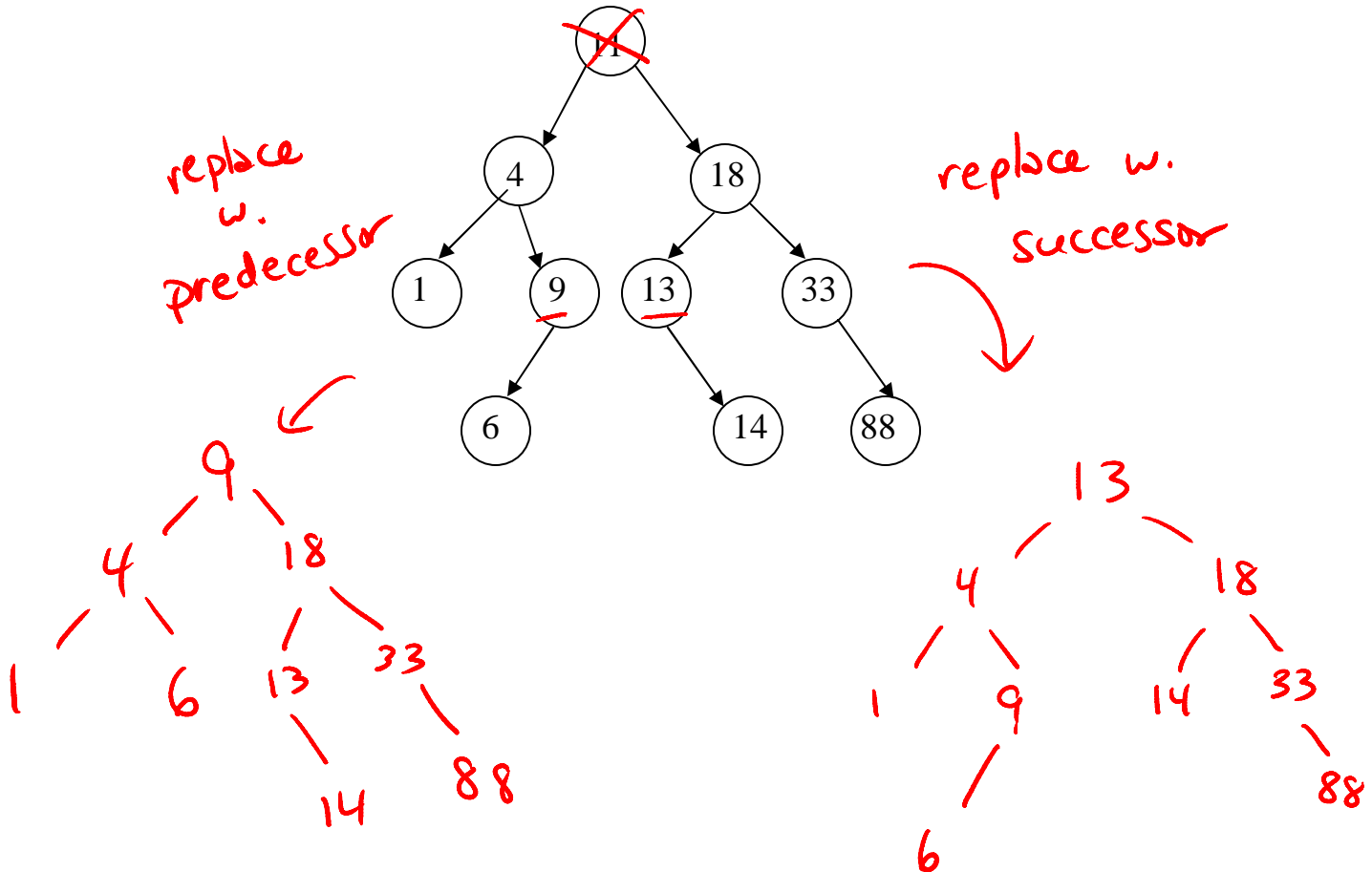
Full: _____a, c_____

Complete: _____a, c_____

AVL balanced: _____a, b, c, d, e, f_____

4. (4 pts total) **Binary Search Trees & AVL Trees**

a) (2 pts) Given the **binary search tree** shown below. Draw what the tree would look like after deleting the value 11. Use one of the methods for deleting described in class or in the book – do NOT use lazy deletion.

replace w. predecessor
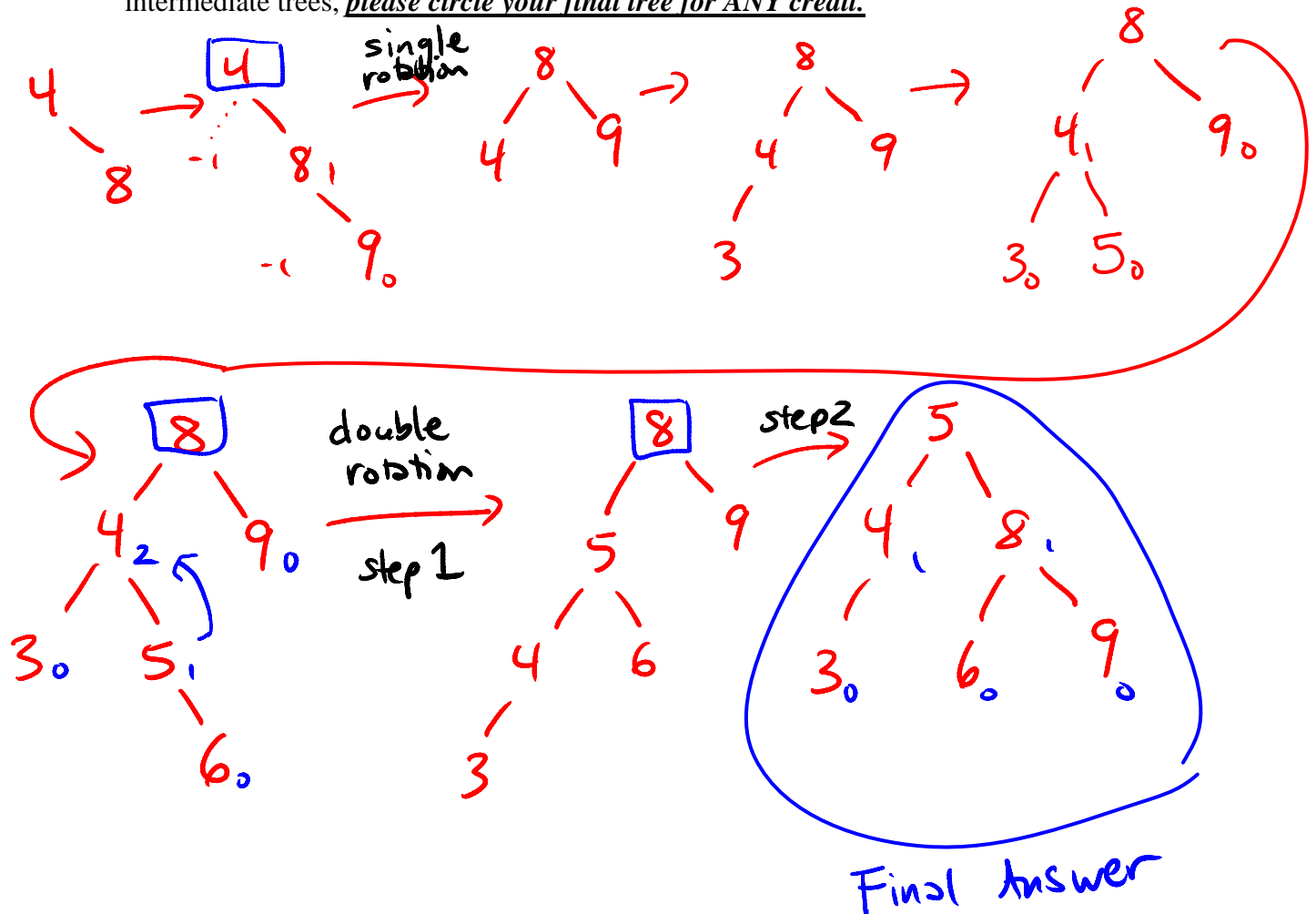
replace w. successor

b) (2 pts) You are given an **AVL tree** of height 6. The minimum and maximum number of *rotations* we might have to do when doing an insert is: (**Give an exact number, not a formula.** A single rotation = 1 rotation, a double rotation = 1 rotation)
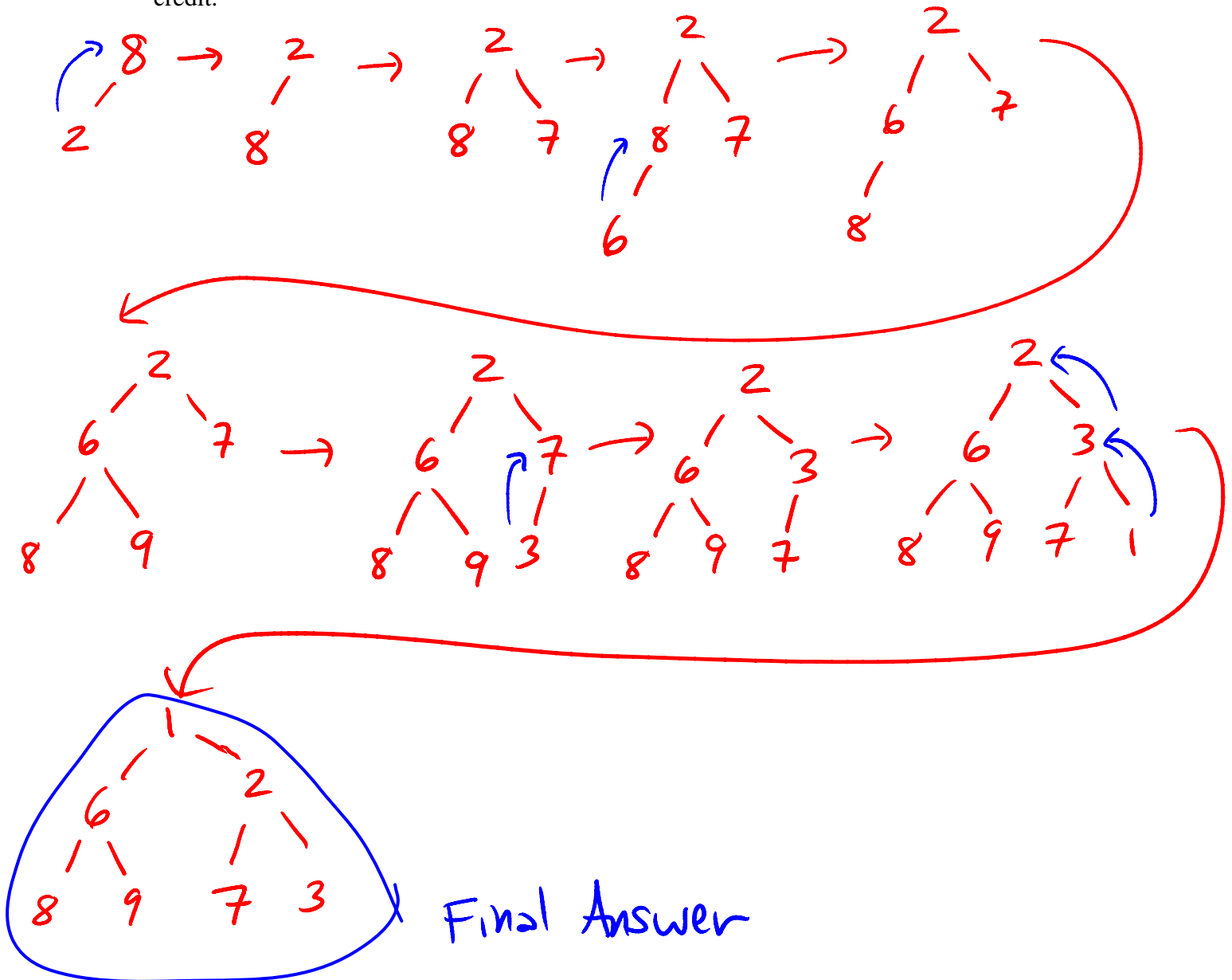
Minimum = 0

Maximum = 1

5. (6 pts) **AVL Trees** Draw the AVL tree that results from <u>inserting the keys:</u>
<u>4, 8, 9, 3, 5, 6</u> in that order into an <u>initially empty AVL tree</u>. 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.__*

(handwritten work showing AVL tree construction with intermediate trees)

single rotation

double rotation

step 1

step 2

Final Answer

6. (8 pts total) **Binary Min Heaps**

(a) [6 points] Draw the binary min heap that results from inserting 8, 2, 7, 6, 9, 3, 1 in that order into an initially empty binary min heap. *You do not need to show the array representation of the heap.* 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 result** for any credit.

8 → 2 → 2 → 2 → 2
2   8   8 7   8 7   6   7
                 6   8

2 → 2 → 2 → 2
6 7   6 7   6 3   6 3
8 9   8 9 3 8 9 7 8 9 7 1

Final Answer

1
6   2
8 9 7 3

6. **(cont.)**

(b) [2 points] Draw the result of one deletemin call on your heap drawn at the end of part (a).

Final Answer
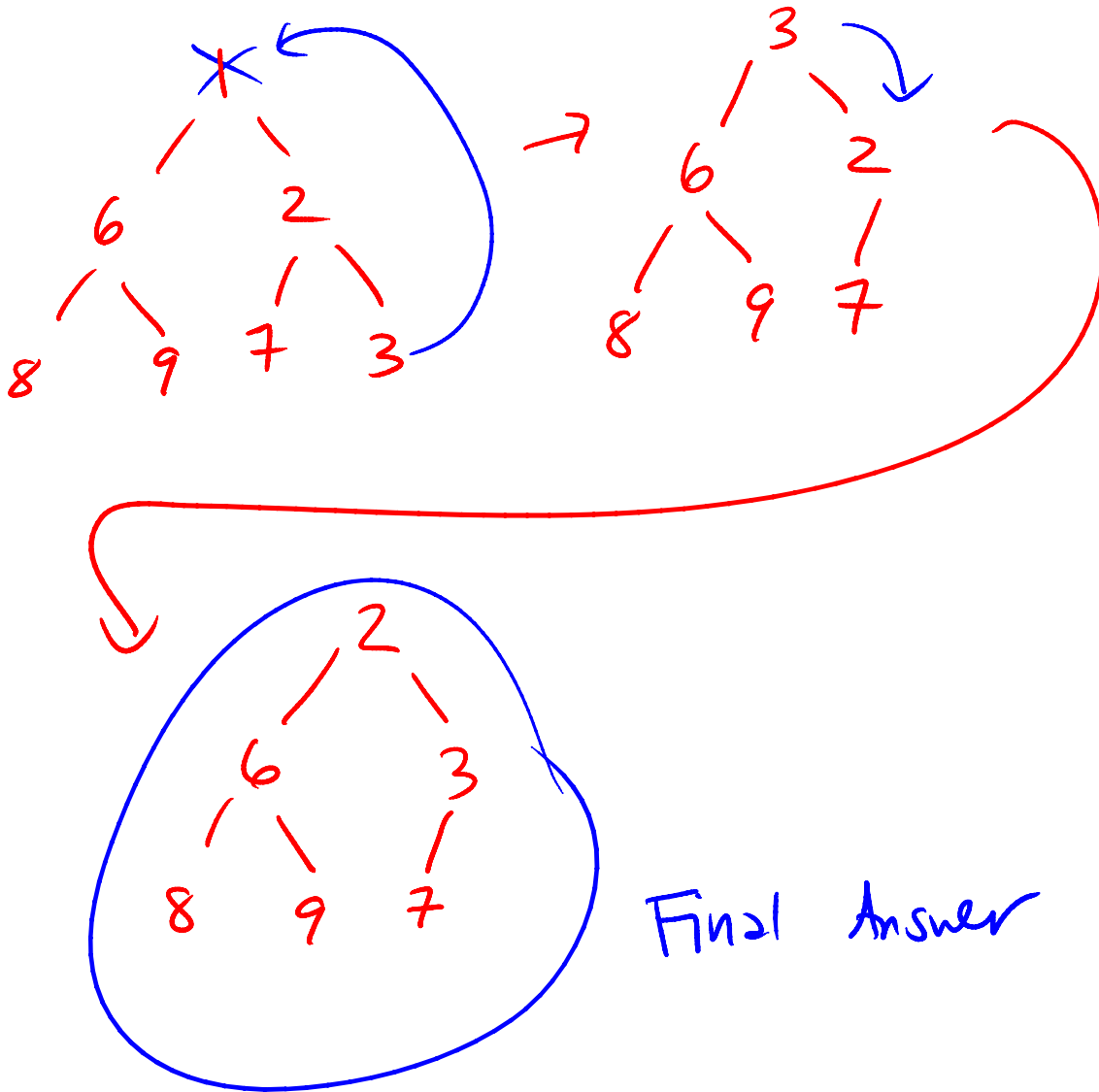
7. (18 pts) **Algorithms & Running Time Analysis:**
- **Describe the most time-efficient way to implement the operations listed below**. Assume no duplicate values and that you can implement the operation as a member function of the class – with access to the underlying data structure.
- Then, give the tightest possible upper bound for the ***worst case*** running time for each operation in terms of *N*. ****For any credit, you must explain *why* it gets this worst case running time.** You must choose your answer from the following (not listed in any particular order), each of which could be re-used (could be the answer for more than one of a) -f)).

$O(N^2)$, $O(N^{1/2})O(N^3 \log N)$, $O(N \log N)$, $O(N)$, $O(N^2 \log N)$, $O(N^5)$, $O(2^N)$, $O(N^3)$, $O(\log N)$, $O(1)$, $O(N^4)$, $O(N^{12})O(N^N)$, $O(N^6)$, $O(N^8)$, $O(N^9)$

a) Pushing a value onto a stack containing N values, implemented as a linked list. **Explanation:**

① Create a new node p ② set p.next → top ③ set top → P. All of these are constant time operations.

a)
$O(1)$

b) Enqueue a value onto a queue containing N values implemented as a circular array (as described in class). (Assume the array is size N+5.) **Explanation**:

① Put new value in array[back]
② back = (back+1) % (N+5)
(All constant time operations.)

front back

b)
$O(1)$

c) Deleting the minimum value in a binary min heap of size N. **Explanation:**

① Remove the root, ② Put rightmost value on bottom row into root position, ③ Percolate down — comparing vs. children + swap w. smaller child if needed. Height = $O(\log N)$ = max # of swaps

c)
$O(\log N)$

d) Given a binary search tree containing N integers, create an AVL tree containing the same values. You should not destroy the original BST in the process. **Explanation:**

Traverse the BST (in any order), as you visit a node, insert that value into the AVL tree. Each AVL insert is $O(\log N)$ due to max height of tree. You are doing N inserts so $O(N\log N)$.

d)
$O(N\log N)$

e) Printing out the values stored in all of the *leaves* of a perfect BST containing N values in ascending order. **Explanation:** Do an inorder traversal, at each node, if both of its children are null, print it out. Checking for null is constant time operation. Must visit each node once → $O(N)$ for traversal.

e)
$O(N)$

f) Given an AVL tree containing N positive integers, print out all the even values contained in the tree in *descending* order (e.g. 12, 8, 6, 2). Be sure to explain how you will get descending order. **Explanation:** Do a traversal as follows: Visit each node once, $O(N)$.

trav(n node){
  trav(right)
  if (n%2==0) print n
  trav(left)
}

f)
$O(N)$