Name: Sample Solution

Email address: _____

CSE 373 Spring 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: 78 points. Time: 50 minutes.

Question	Max Points	Score
1	16	
2	8	
3	14	
4	6	
5	6	
6	10	
7	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)):

$O(N^2)$, $O(N^{½})O(N^3 \log N)$, $O(N \log N)$, $O(N)$, $O(N^2 \log N)$, $O(N^5)$, $O(2^N)$, $O(N^3)$, $O(\log N)$, $O(1)$, $O(N^4)$, $O(N^{12})O(N^N)$, $O(N^6)$, $O(N^8)$, $O(N^9)$

You do not need to explain your answer.

a) *f(N)* = $N \cdot (N^2 \log N + N^2)$ $O(N^3 \log N)$

b) *f(N)* = $\log_{16}(2^N)$ $O(N)$

c) *f(N)* = $(N/4) \log (N/4) + N/4$ $O(N \log N)$

d) *f(N)* = $(2N + 2N)^3$ $O(N^3)$

e) *f(N)* = $N^{½} + \log N$ $O(N^{\frac{1}{2}})$

f) *f(N)* = $N \log (100^3)$ $O(N)$

g) *f(N)* = $(N \cdot N \cdot N \cdot N)^3$ $O(N^{12})$

h) *f(N)* = $(N^3 + 2N) / N$ $O(N^2)$

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

II.
```
int silly(int n, int m) {
   if (n < 1) return n;
   else if (n < 100)
     return silly(n - m, m);
   else
     return silly(n - 1, m);
}
```
$O(n)$

III.
```
void silly(int n) {
   j = 0;
   while (j < n) {
      for (int i = 0; i < n; ++i) {
         System.out.println("j = " + j);
      }
      j = j + 5;
   }
}
```
$O(n^2)$

IV.
```
void silly(int n) {
   for (int i = 0; i < n * n; ++i) {
      for (int j = 0; j < n; ++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);
      }
   }
}
```
$O(n^5)$

3. (14 pts total) **Trees**.

a) (3 pts) What is the minimum and maximum number of nodes *at depth d* in a **full binary tree**? Be sure to list the nodes **at** depth d. Do not include nodes at depth d-1 or d+1 or other depths. (Hint: the root is at depth 0)

Minimum = 2 (if height ≥ 1, otherwise 1)

Maximum = 2^d

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

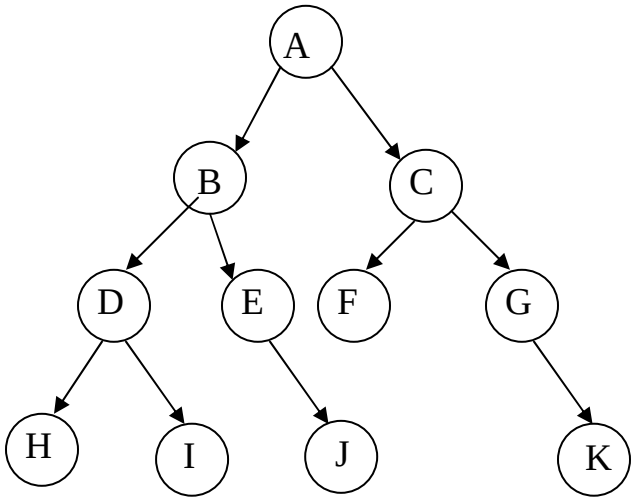
Pre-Order: A B D H I E J C F G K

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

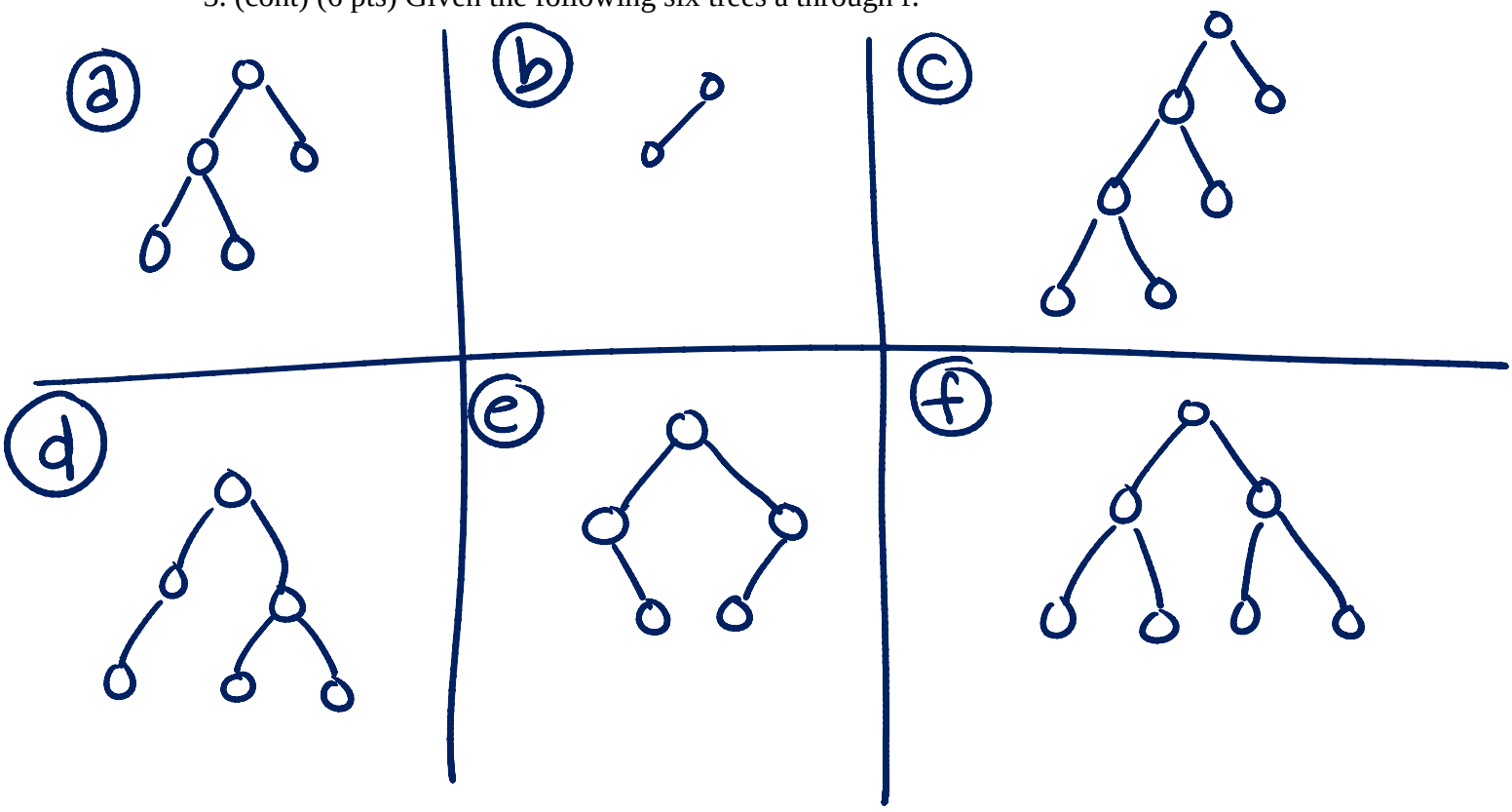
In-Order: H D I B E J A F C G K

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

d) (1 pt) Is it AVL balanced (ignore the values, only look at the shape): YES / NO



3. (cont) (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, f

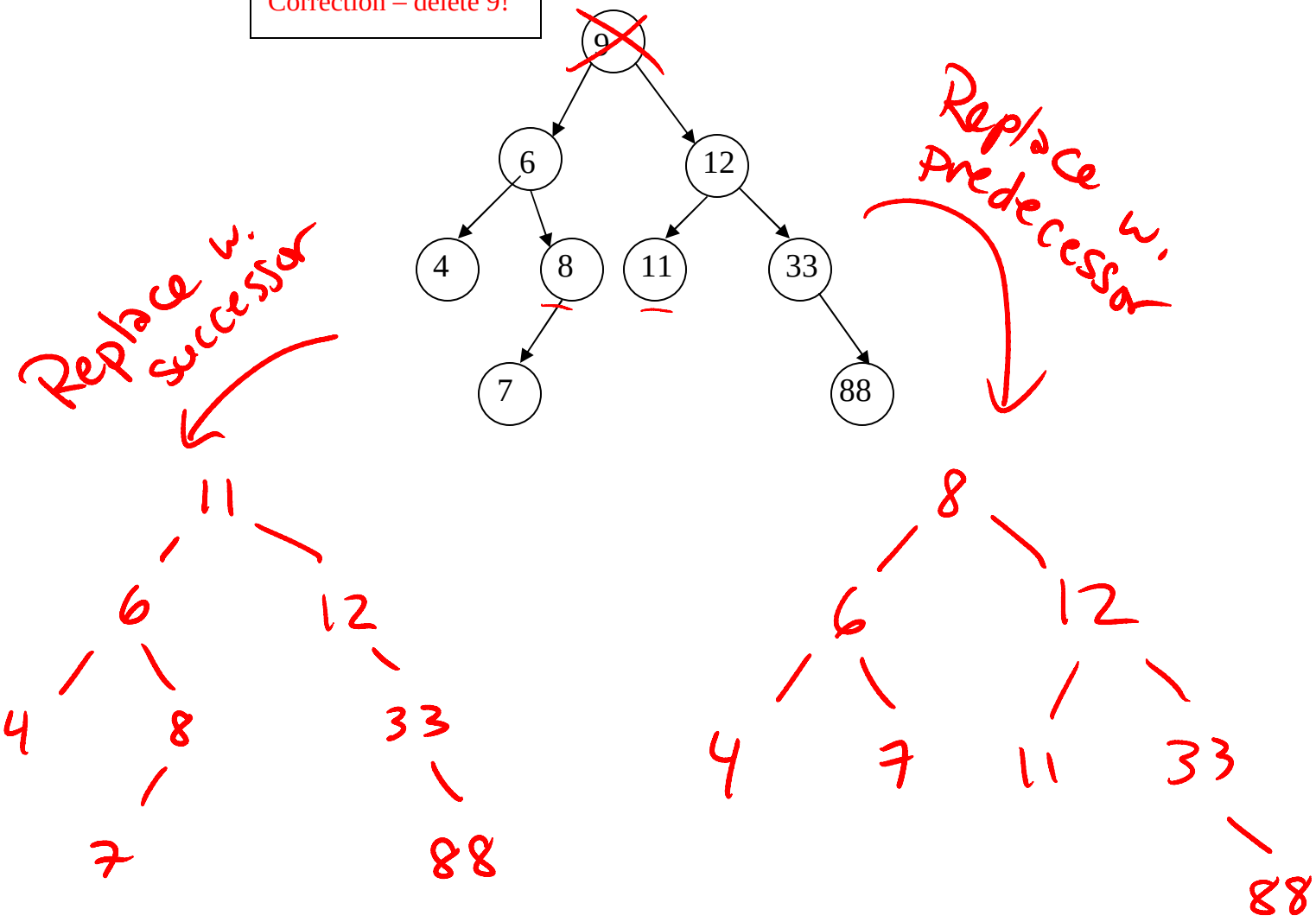
Complete: a, b, f

AVL balanced: a, b, d, e, f

4. (6 pts total) **Binary Search Trees & Binary Min Heaps**

a) (3 pts) Given the **binary search 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.

Correction – delete 9!

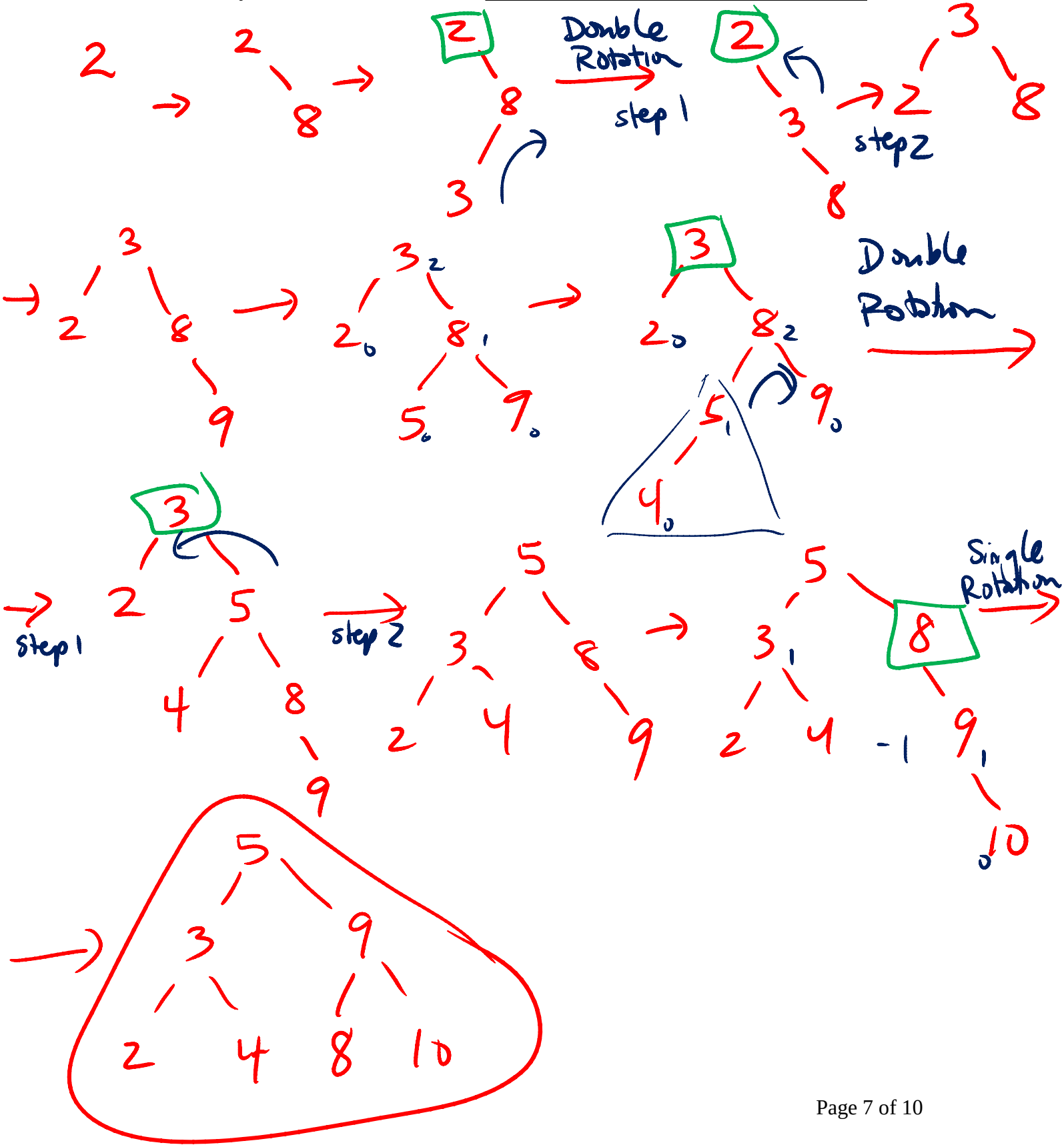


b) (3 pts) You are given a **binary min heap** of height 6. The minimum and maximum number of *comparisons* we might have to do when doing a deletemin is:

Minimum = 2

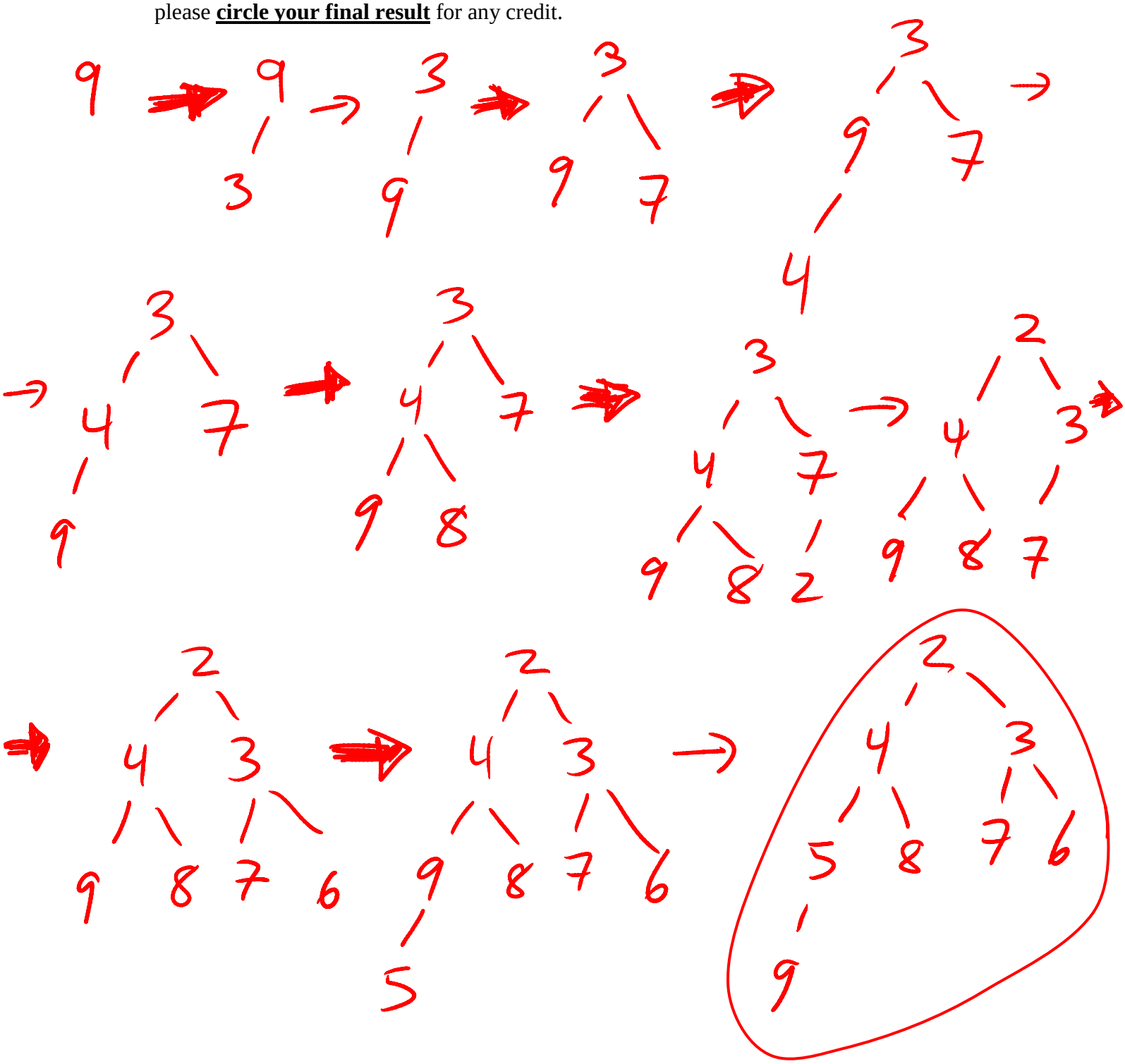
Maximum = 12

5. (6 pts) **AVL Trees** Draw the AVL tree that results from inserting the keys: 2, 8, 3, 9, 5, 4, 10 in that order into an initially empty AVL 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.***



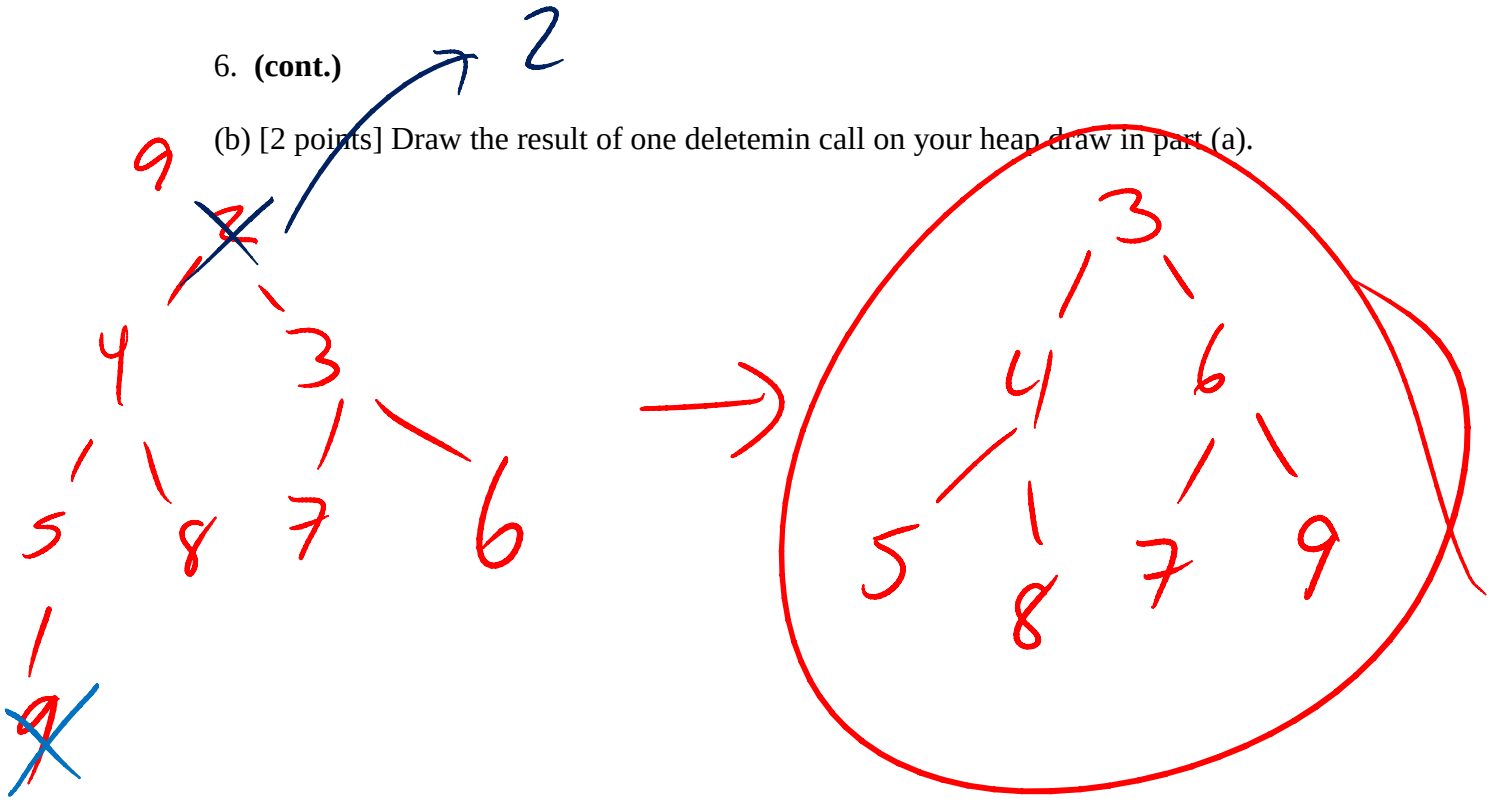
6. (10 pts total) **Binary Min Heaps**

(a) [6 points] Draw the binary min heap that results from inserting ~~9, 3, 7, 4, 8, 2,~~ 6, 5 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.

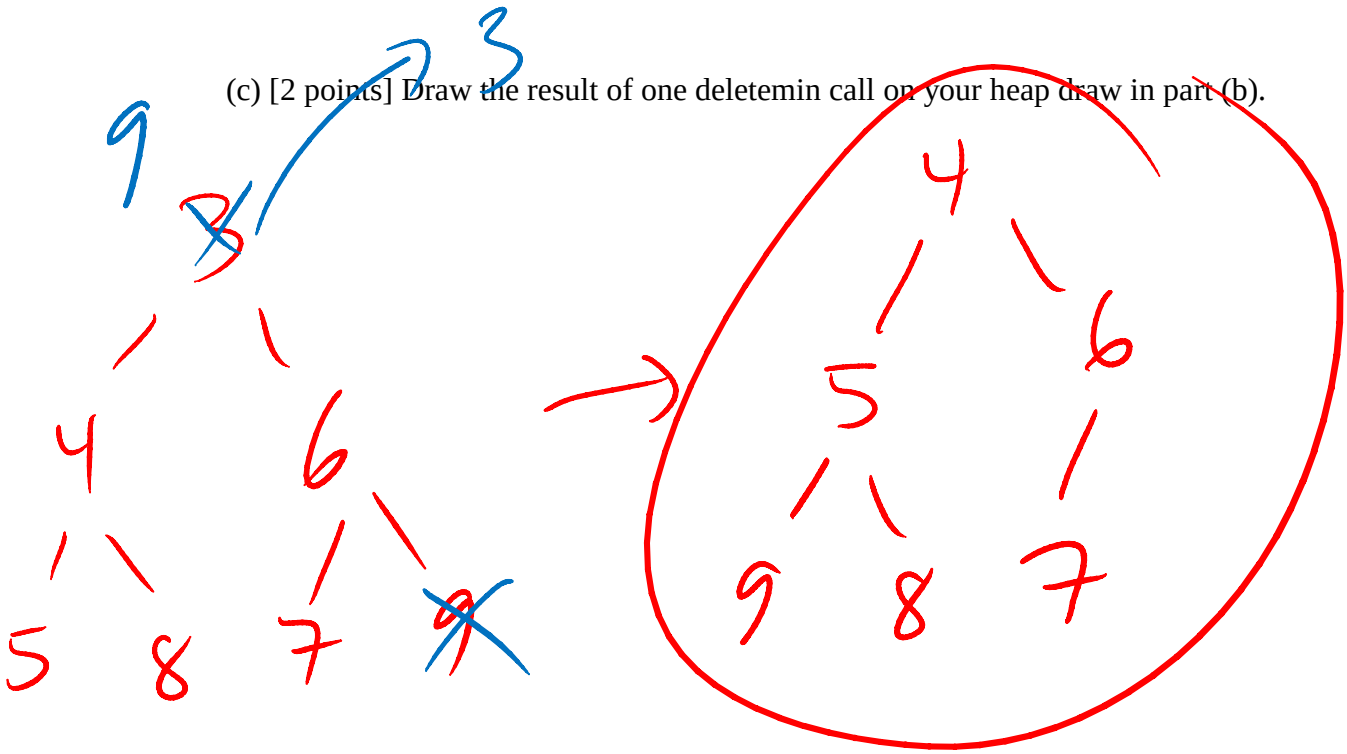


6. **(cont.)**

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



(c) [2 points] Draw the result of one deletemin call on your heap draw in part (b).

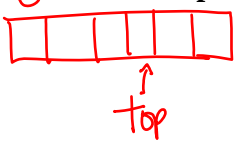


7. (18 pts) **Running Time Analysis:** Give the tightest possible upper bound for the ***worst case*** running time for each of the following in terms of *N*. You MUST choose your answer from the following (not given in any particular order), each of which could be re-used (could be the answer for more than one of a) – f)):

$O(N^2)$, $O(N^{½})O(N^3 \log N)$, $O(N \log N)$, $O(N)$, $O(N^2 \log N)$, $O(N^5)$, $O(2^N)$, $O(N^3)$, $O(\log N)$, $O(1)$, $O(N^4)$, $O(N^{12})O(N^N)$, $O(N^6)$, $O(N^8)$, $O(N^9)$

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

a) Pop a value off a stack containing N elements implemented as an array.
Explanation: Grow stack from index 0 → size-1. So popping just involves returning array[top] & decrementing top. Both are constant time operations.



a) O(1)

b) Printing out all the odd values stored in a binary search tree containing N positive integers *in ascending order*. **Explanation:** Do an in-order traversal (O(N)) & for each node, if it's value is odd, print it out. Checking if odd & printing are both constant time operations – do them once per node.

b) O(N)

c) Finding the minimum value in a binary search tree of size N. **Explanation:** To find the minimum, go left until you hit a null ptr. In the worst case, the tree could look like: so you would have to visit all N nodes.

direction matters, Note: This is an easy case

c) O(N)

d) Moving the values from a binary min heap, into an initially empty array of the same size. The final contents of the array should be sorted from low to high. **Explanation:** Call deletemin N times. Each call to deletemin gives you the next value (from low to high) & costs O(log N), write it to next empty location in array is constant time. So cost is N ops of cost O(log N) = O(N log N).

d) O(N log N)

e) Finding the *maximum* value in a binary min heap of size N. **Explanation:** The max will be one of the last N/2 values, so you can just search those. But this is still O(N).

e) O(N)

f) Printing out the values in an AVL tree in post-order. **Explanation:** A post-order traversal visits each node once regardless of the shape of the tree. So O(N) for AVL tree or regular BST.

f) O(N)