CSE 373, Spring 2012 Midterm Solutions

0. The First Question (0 points)

What is the answer to this first question?

e

1. Big-Oh (5 Points)

```
public int add100(int[] array) {
    if (array.length < 100) {
        return 0;
    }

    int sum = 0;
    for (int i = 0; i < 100; i++) {
        sum += array[i];
    }
    return sum;
}
```

What is the running time of the method above? Explain your answer.

O(1). For large n, the algorithm only adds up the first 100 elements. The running time does not depend on n.

General rubric:
1 Indicates constant time
4 Explanation (pretty much all or nothing)

Common errors:
- Saying that the method is O(n) up to 100 elements—this is not Big-Oh is about (-2)

2. Summations / Big-Oh (15 Points)

Calculate the approximate value of the variable `sum` after the following code fragment, in terms of variable `n`. Use summation notation to compute a closed-form solution (ignore small errors caused by `i` not being evenly divisible by 2). Then use this value to give a tightly bounded Big-Oh analysis of the runtime of the code fragment.

```
int sum = 0;
for (int i = 1; i <= 10; i++) {
    for (int j = n; j <= n + i + 2; j++) {
        sum++;
        sum++;
    }
}

for (int i = 0; i < n; i += 2) {
   sum++;
}
```

$$\left(\sum_{i=1}^{10}\sum_{j=n}^{n+i+2} 2\right) + \sum_{i=1}^{\frac{n}{2}} 1$$

$$\left(\sum_{i=1}^{10} 2(n+i+2-n+1)\right) + \frac{n}{2}$$

$$\left(\sum_{i=1}^{10} 2\cdot(i+3)\right) + \frac{n}{2}$$

$$\sum_{i=1}^{10}(2\cdot i+6) + \frac{n}{2}$$

$$\sum_{i=1}^{10} 2\cdot i + \sum_{i=1}^{10} 6 + \frac{n}{2}$$

$$2\sum_{i=1}^{10} i + 60 + \frac{n}{2}$$

$$2\frac{10(11)}{2} + 60 + \frac{n}{2}$$

$$110 + 60 + \frac{n}{2}$$

$$170 + \frac{n}{2}$$

Big-Oh: O(n)

Common errors:
- Having the inner summation evaluate to `i+2`
- Not substituting 10 into n*(n+1) / 2 resulting in O(n^2)
- Losing the factor of 2 somehow

General rubric:
5 Having summation notation
5 Answer (-3 first mistake, -2 additional mistake)
5 Big-Oh

3. **Sorting** (15 Points)

Consider the follow array:

```
[38, -3, 47, 55, 1, 58, 16, 96, -84, 7]
```

Each of the following is a view of a sort in progress of the above array. Which sort is which? Each sort is used exactly once.

- Choose between **bubble sort**, **selection sort**, **insertion sort**, **merge sort**, and **quick sort**.
- If the sorting algorithm contains multiple loops, the array is shown after a few of passes of the outermost loop has completed.
- If the sorting algorithm is **merge sort**, the array is shown after the recursive calls have completed on each sub-part of the array.
- If the sorting algorithm is **quick sort**, the algorithm chooses the *first* element as its pivot. For quick sort, the array is shown before the recursive calls are made.

a. `[-84, -3, 1, 55, 47, 58, 16, 96, 38, 7]`

Sorting Algorithm: **selection sort**

b. `[-3, 1, 38, 16, 47, -84, 7, 55, 58, 96]`

Sorting Algorithm: **bubble sort**

c. `[7, -3, -84, 16, 1, 38, 55, 96, 47, 58]`

Sorting Algorithm: **quick sort**

d. `[-3, 1, 38, 47, 55, -84, 7, 16, 58, 96]`

Sorting Algorithm: **merge sort**

e. `[-3, 38, 47, 55, 1, 58, 16, 96, -84, 7]`

Sorting Algorithm: **insertion sort**

Common errors:
- Swapping bubble sort with merge sort, since top half of bubble sort is sorted (but bottom half isn't!)

General rubric:
3 per answer (all or nothing)

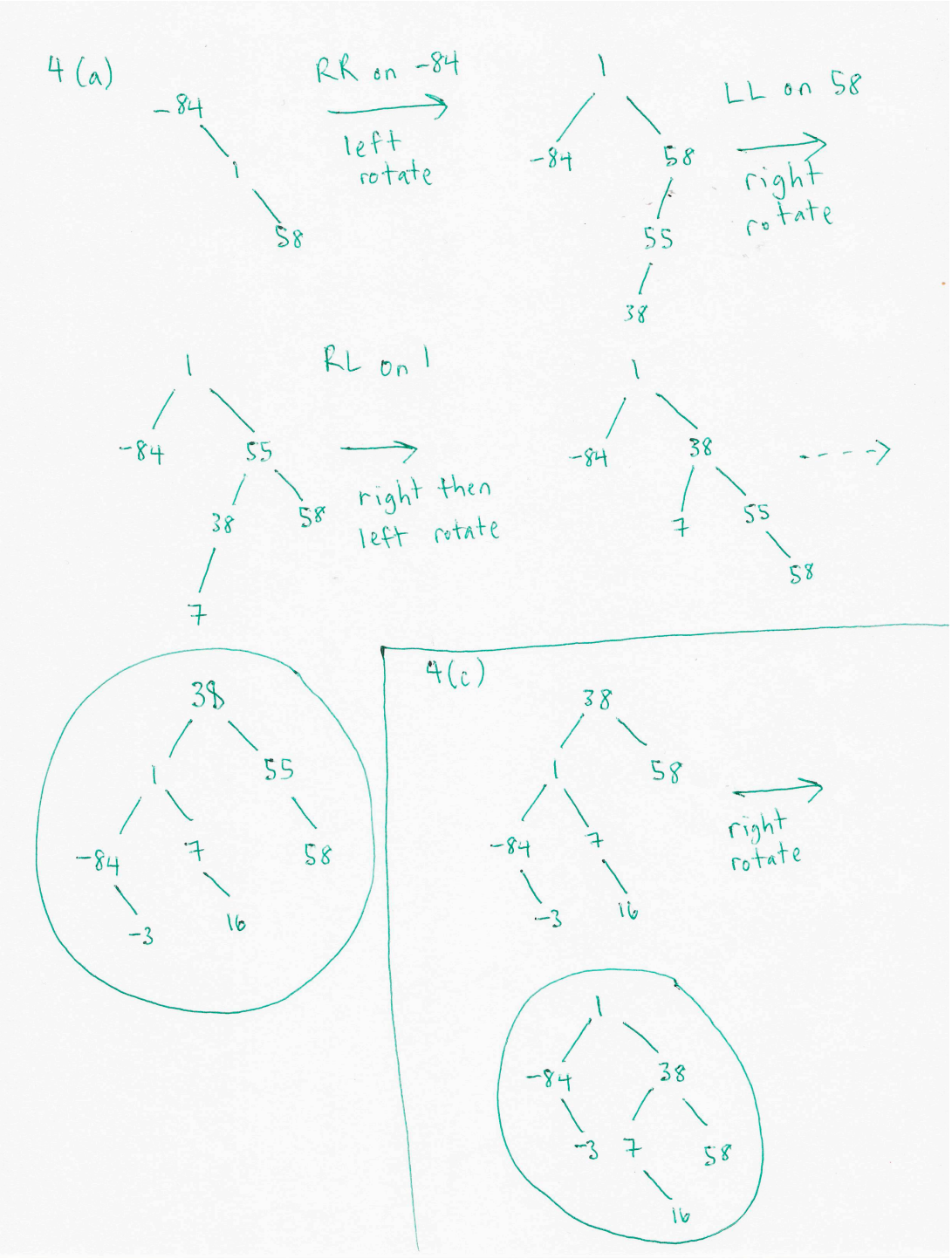
4. AVL Trees (20 Points)

a. Given the following list of integers:

-84, 1, 58, 55, 38, 7, -3, 16

Draw the **AVL tree** that results when all of the above elements are added (in the given order) to an initially empty AVL tree.

Please show your work. You do not have to draw an entirely new tree after each element is added, but since the final answer depends on every add being done correctly, you may wish to show the tree at various important stages to help earn partial credit in case of an error. The next page is blank to give you space to write.



a.

Common errors:
- Incorrect rotation
- Inserting element randomly into the wrong spot

General rubric:
5 has balanced BST with everything
5 correct (-3 first mistake, -2 for additional mistakes)

b. What is the balance factor of the root node of the **AVL tree** that you drew for part a?

-1

Common errors:
- Confusing the left and right subtrees

General rubric:
4 correct based on tree in part a

c. Draw the resulting **AVL tree** after you remove 55 from your tree in part a.

Common errors:
- Incorrect rotation
- Inserting element randomly into the wrong spot

General rubric:
3 tree is a balanced BST without 55
3 correct

5. **Heaps** (20 Points)

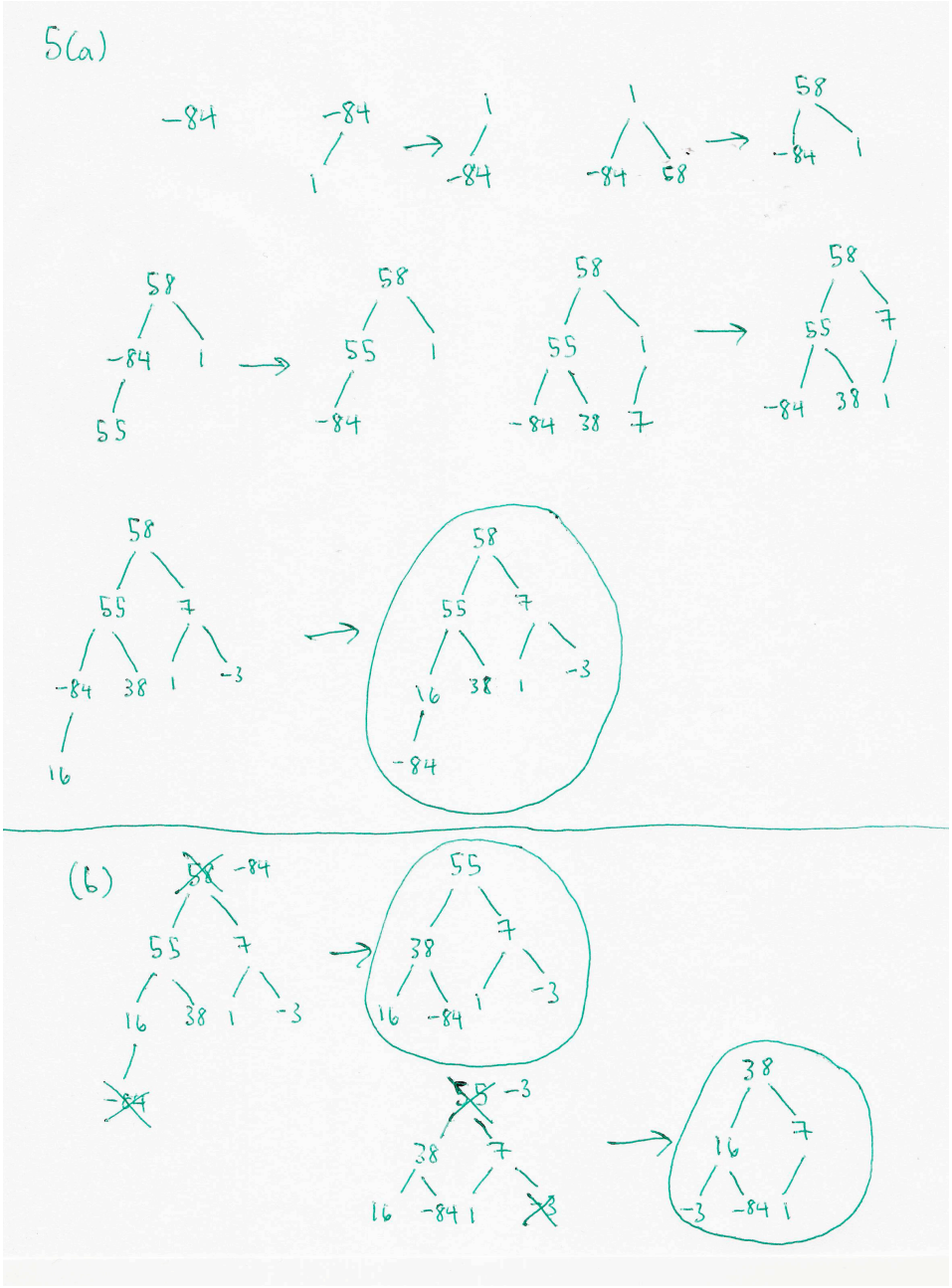
Given the following integer elements:

-84, 1, 58, 55, 38, 7, -3, 16

a. Draw the tree representation of the heap that results when all of the above elements are added (in the given order) to an initially empty **maximum** binary heap. ***Circle the final tree that results from performing the additions.***

b. After adding all the elements, perform **two (2) removes** on the heap. ***Circle the tree that results after the two elements are removed***.

Please show your work. You do not need to show the array representation of the heap. You do not have to draw an entirely new tree after each element is added or removed, but since the final answer depends on every add/remove being done correctly, you may wish to show the tree at various important stages to help earn partial credit in case of an error. The next page is blank to give you space to write.



a. Common errors:
- Doing a min heap instead of a max heap (-5)
- Moving a random value to the top instead of the last value

General rubric:
10 complete and a heap
5 correct (-3 per error)

b. Common errors:
- Not switching with the larger child during a bubble down
- Removing random values instead of the largest value
- Moving a random value to the top instead of the last value

General rubric per removal:
3 complete and a heap
2 correct

6. Heap Programming (20 Points)

a. In lecture, we implemented a class called `IntBinaryHeap`, a minimum binary heap for integers. Add a method to this class called `findMax` that accepts no parameters and returns the maximum value in the heap. In the figure below, the method would return `39`. If the heap is empty, throw a `NoSuchElementException`.

(Yes, this method is silly to have in a minimum binary heap.)

The class declaration of `IntBinaryHeap` along with its fields is given below. You should write `findMax` as if you were adding it to this class. Recall that we implemented `IntBinaryHeap` with an array data structure (represented by the `array` field) and that the root (i.e. the minimum value) was stored at index 1. For an example of this representation, see **Error! Reference source not found.** and **Error! Reference source not found.** (omitted in solutions).

```
public class IntBinaryHeap implements IntPriorityQueue {
    private static final int DEFAULT_CAPACITY = 10;
    private int[] array;
    private int size;

    ...

    // THE findMax METHOD GOES HERE
    public int findMax() {
        if (isEmpty()) {
            throw new NoSuchElementException();
        }

        int max = array[size];
        int i = size;
        while (i * 2 > size) {
            if (array[i] > max) {
                max = array[i];
            }
            i--;
        }

        return max;
    }
}
```

I forgot to put a condition that the method search as few nodes as possible, so most people went through the entire heap.

```
    public int findMax() {
        if (isEmpty()) {
            throw new NoSuchElementException();
        }

        int max = array[1];
        for (int i = 2; i <= size; i++) {
            if (array[i] > max) {
                max = array[i];
        }

        return max;
    }
```

Common errors:
- not including `size` in the loop bounds (-2)
- assuming the max can start off at 0—what if all numbers are negative? (-3)
- searching only bottom-most level—this is not valid (-3)
- using `array.length` instead of `size` (-2)
- traversing heap as a tree but only going down one path (-7)

General rubric:
3 header (-1 for each of not public, has static, wrong return type/parameters)
3 empty heap (2 for condition, 1 for throwing exception)
4 loop (2 proper loop, 2 loop bounds)
2 returning something
3 all correct

b. What is the worst-case runtime of your `findMax` method? Give your answer in Big Oh notation. Explain your answer.

O(n). If the tree is full, then the algorithm would have to go through all the leaves at the bottom which make up half the nodes. There are O(n/2) leaves.

Your explanation had to more-or-less align with your implementation.

Common errors:
- Saying that searching the bottom-most level is O(log n). The bottom-most level can have up to n/2 elements. It is still O(n)

General rubric:
3 Big-Oh
2 Explanation

7. **Miscellaneous** (5 Points)

a. What generally causes a `StackOverflowException`? Why does it happen?

Infinite recursion. When a method calls another method, information about the calling method is put onto a stack, so you know where to return to after the called method is finished. If the method is recursive and continuously recurses, then all the method calls have to be saved. At some point, the stack becomes full and you get a stack overflow exception.

General rubric:
2 Something along these lines. We were generally generous unless you said something really wrong.

b. The tightest-bound worst case running time of algorithm A is $O(n^2)$. The tightest-bound worst case running time of algorithm B is O(n log n). Is there any reason to use algorithm A? Explain your answer.

- It is possible that the *average-case* running time of algorithm A is better than the *average-case* running time of algorithm B. This is the case with quick sort.

- For smaller *n*, algorithm A might actually be faster than algorithm B. This is the case with insertion sort.

- The input data might have specific properties (e.g., mostly ascending) that make it more suitable for algorithm A. This is the case with insertion sort.

General rubric:
3 Something along these lines (didn't have to list all reasons). We were generally generous unless you said something really wrong.