CSc 127B — Introduction to Computer Science II
Fall 2015 (McCann)

Answers to the Practice Questions ahead of the Final Exam

The section leaders were kind enough to write some practice questions for you to work as you prepare for the exam; those are in another handout. These are the SLs' suggested answers for the questions.

1.

(a)

```
          7
        /   \
      4       8
    /   \
  2       3
```

(b) 2, 3, 8

(c) 7

(d) 2

(e) 3

2.

(a)

```
          9
        /
      5
    /
   4
  /
 3
```

(b) 0

(c) 1

3.

(a) O(n) (loop iterates n times over a constant # of statements)

(b) O(n^2) (this is the sort of loop nesting you'd see in Selection Sort or Insertion Sort; both loops iterate a max of n time; $n * n = n^2$)

(c) O($\log_2 n$) (we're counting toward n by powers of 2, so it will require approximately $\log_2 n$ iterations to get there)

(d) O(1)

(e) O(n)

(Continued . . .)

4.

(a)

1 4 6 9 2 3 7

1 2 6 9 4 3 7

1 2 3 9 4 6 7

1 2 3 4 9 6 7

1 2 3 4 6 9 7

1 2 3 4 6 7 9

(b)

1 8 7 6 5 4 3 2 9

1 2 7 6 5 4 3 8 9

1 2 3 6 5 4 7 8 9

1 2 3 4 5 6 7 8 9

1 2 3 4 5 6 7 8 9

1 2 3 4 5 6 7 8 9

1 2 3 4 5 6 7 8 9

5. The algorithm doesn't have the ability to notice that the list is already sorted. It must continue through all of the passes in all cases, best, worst or average.

6.

(a)

4 7 6 8 2 3 (1 key comparison(s), 3 copies)

4 6 7 8 2 3 (2 key comparison(s), 3 copies)

4 6 7 8 2 3 (1 key comparison(s), 0 copies)

2 4 6 7 8 3 (4 key comparison(s), 6 copies)

2 3 4 6 7 8 (5 key comparison(s), 6 copies)

Totals: 13 key comparisons, 18 data movements

(b)

7 8 6 5 4 3 (1 key comparison(s), 3 copies)

6 7 8 5 4 3 (2 key comparison(s), 4 copies)

5 6 7 8 4 3 (3 key comparison(s), 5 copies)

4 5 6 7 8 3 (4 key comparison(s), 6 copies)

3 4 5 6 7 8 (5 key comparison(s), 7 copies)

Totals: 15 key comparisons, 25 data movements

(c) This sequence is already sorted. Each item only gets compared to the item at the end of the list (there are n-1 passes, so n-1 comparisons), and no data to copy.

Totals: 5 key comparisons, 0 data movements

(Continued . . .)

7.

(a)

i.

```
     5
   /   \
  3       8
 / \     / \
2   4   7   9
```

ii.

```
1
 \
  2
   \
    3
     \
      4
       \
        5
         \
          6
           \
            7
```

(b) The second one. Each insertion was deeper than the last, resulting in a tree than looks like a linked list. We know that searching to the end of a linked list is O(n), but searching to the bottom of a ‘bushy’ tree is just $O(\log_2 n)$.

8.

(a) The answer to this depends on how the Quicksort algorithm is implemented. If we assume that Quicksort is called on any size list, then there will be 7 total calls: The call to sort the original list, the call to sort the list of one item (a base case, so it returns immediately), the call to sort the right side (7 4 5 8), the call to sort 7 4 5, the call to sort 4, the call to sort 7, and the call to sort the length zero list to the right of 8.

(b) Divide and Conquer

(c) Choosing the largest item in the list, or the smallest item in the list, as your pivot. In either situation, the list shrinks by only one item instead of cutting the list in half, and Quicksort can sort two half-lists a lot faster than one list with one less item.

9.

	Class	Package	Subclass	World
public	Y	Y	Y	Y
protected	Y	Y	Y	N
-	Y	Y	N	N
private	Y	N	N	N

(I didn’t actually have time to present this in class, but I did post slides on it. –lim)