CS 112—Midterm Exam—Spring 2014

There are 8 problems on the exam. The first and last are mandatory, and you may eliminate any one of problems 2 – 7 by drawing an X through them. Problem 1 is worth 10 points, and all other problems are worth 15. Please write in pen if possible. Circle answers when they occur in the midst of a bunch of calculations. If you need more room, use the back of the sheet and tell me this on the front sheet.

Problem One. (True/False – MANDATORY – 10 points) Write True or False to the left of each statement. Red are false, black are true.

1. Mergesort is difficult to implement “in place” (i.e., without using an extra array).

2. A class can never be a private member of another class.

3. The predefined class Integer is a wrapper class that enables you to use an int as if it were a class.

4. The principle of Step-Wise Refinement is that you should break your problem (and its solution) into manageable-sized pieces.

5. If a class in Java implements an interface, but also implements a public method not defined in the interface, it will cause a compile-time error.

6. If f(n) = Θ(g(n)) and g(n) = Θ(h(n)) then f(n) = Θ(h(n))

7. The worst case time of quicksort is the same (in terms of Θ(....)) as the best case time of selection sort.

8. “Tail recursion” refers to the situation where the recursive call occurs last in the recursive case, and there is no base case.

9. The maximal node in the right subtree of the root may be found by starting at the root and continuing to follow the right pointer until you can go no further.

10. KISS stands for “Kode It Soon, Stupid”
Problem Two (Simulation and Analysis -- 15 points). This has two parts. (A) Perform Quicksort on the following array, according to the algorithm presented in lecture; if you get to a sub-problem of size 2, you may leave it as is (if they are in the right order) with no comparisons, or exchange the two (if they are not in the right order) and count 1 comparison. Show the array after every exchange and underline each pivot value.

3, 9, 8, 4, 6, 1, 0, 2

(B) Answer the following questions in a short sentence or two: What is the worst-possible choice for a pivot value during the partition step and what is the time complexity of the algorithm when this happens every time? What is the best possible choice and what is the time complexity when this happens every time?

A

3, 9, 8, 4, 6, 1, 0, 2
3, 2, 8, 4, 6, 1, 0, 9
3, 2, 0, 4, 6, 1, 8, 9
3, 2, 0, 1, 6, 4, 8, 9
1, 2, 0, 3, 6, 4, 8, 9
1, 0, 2, 3, 6, 4, 8, 9
0, 1, 2, 3, 6, 4, 8, 9
0, 1, 2, 3, 4, 6, 8, 9

B

The worst-possible pivot is the minimum or maximum value, which results in a $\Theta(N^2)$ time; the best-possible is the median value, which results in a $\Theta(N \log N)$ time.
Problem Three (Recursion on Linked Lists – 15 points). Consider this linked list:

```
public static Node conundrum(Node p) {
    if(p == null || p.next == null)
        return p;
    else {
        Node q = p.next;
        p.next = conundrum(q.next);
        q.next = p;
        return q;
    }
}
```

// Apply the method to the linked list as follows:

```java
head = conundrum(head);
```

Solution: Same as problem 3 on the sample final: see the solution posted there!
Problem Four (Binary Search Trees--15 points). Perform the following operations on an initially empty binary search tree, show the tree that would result at the end of each of each insertion or deletion. Note: when deleting from the tree, always go to the right to find the node to replace the deleted node.

(a) Insert the following values into an initially empty tree: 6, 2, 18, 9, 11, 12, 8, 25, 40. Show the tree that would result.

(b) From the following tree, delete the value at the root four times by alternately deleting from the right and the left (i.e., R, L, R, L).

(c) Supposing that you may only put integers into the tree, and can not have duplicates, what are the only values that can be inserted to the left of the node containing 25 in the tree in part (b)?

Solution: 21, 22, 23, 24
Problem Five (Trace an algorithm -- 15 points). Consider the following example of a linked list:

![Linked List Diagram]

Apply the following recursive algorithm to this list, and show what it prints out.

```java
void mystery(Node p) {
    if (p == null) {
    } else {
        System.out.println(p.item);
        if (p.item > 5) {
            mystery(p.next);
        } else {
            mystery(p.next);
            mystery(p.next);
        }
    }
}
```

SOLUTION:

6
1
8
2
8
2
Problem Six (Write an algorithm -- 15 points). Suppose you have a queue data structure which satisfies the following interface:

```java
public interface StringQueue {
    public void enqueue(String s);
    public String dequeue();
    public int size();
    public boolean sameEnds();
}
```

You have implemented all but the last method, which returns true if the elements at the front and the rear of the queue are equal (e.g., “hi” and “hi”), and false if the queue has fewer than two elements or if the front and rear elements are not equal.

Show how to implement `sameEnds()` using ONLY the other three methods; you may declare String variables but may not use an auxiliary array or create another queue. You may use any control structure you like, e.g., loops or recursion.

SOLUTION:

```java
public boolean sameEnds() {
    if(size() < 2)
        return false;
    String first = dequeue();
    String last;
    while(size() > 0) {
        last = dequeue();
    }
    return (first.equals(last));
}
```
Problem Seven (Asymptotic Analysis -- 15 points). How many times does the following code print out “Hi there!”? Express your answer as a function of N, in terms of $\Theta(\ldots N \ldots)$.

```java
for( int i = 0; i < N; i = i + 2 )  // $\Theta(N)$
    for( int k = N; k >= 1; k = k / 2 )  // $\Theta(\log N)$
        for( int j = 1; j < 1000; j = j * 2 )  // $\Theta(1)$
            for( int m = N; m > -N; --m )  // $\Theta(N)$
                System.out.println("Hi there!");
```

SOLUTION: $\Theta(N^2 \log N)$
Problem Eight. (MANDATORY – 15 points) For the following algorithms, for the first set, state the worst-case time (in terms of $\Theta$) for performing the indicating operation and then state what situation (e.g., what configuration of the data structure and what item) produces this worst-case behavior. For the second set, simply give the $\Theta(...N...)$ estimate. The number of items stored in each data structure is $N$.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Worst Case $\Theta(......)$</th>
<th>Worst Case Situation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Find an item in an ordered array using Binary Search</td>
<td>$\Theta(\log N)$</td>
<td>Try to find minimal or maximal element, or element not in array.</td>
</tr>
<tr>
<td>Insert an item into an unordered list implemented as an array without resizing</td>
<td>$\Theta(1)$</td>
<td>All cases the same!</td>
</tr>
<tr>
<td>Dequeue an item from a queue</td>
<td>$\Theta(1)$</td>
<td>All cases the same!</td>
</tr>
<tr>
<td>Enqueue an item into a queue implemented as a ring buffer with resizing</td>
<td>$\Theta(N)$</td>
<td>Insert causes resizing!</td>
</tr>
<tr>
<td>Insert an item into a Binary Search Tree</td>
<td>$\Theta(N)$</td>
<td>Binary tree is really a linked list, and you insert at end of list, e.g., maximal element.</td>
</tr>
<tr>
<td>Delete an item from a Binary Search Tree</td>
<td>$\Theta(N)$</td>
<td>Binary tree is really a linked list, and you delete at end of list, e.g., maximal element.</td>
</tr>
<tr>
<td>Pop from a stack made from a linked list.</td>
<td>$\Theta(1)$</td>
<td>All cases the same.</td>
</tr>
<tr>
<td>Add an element in the exact middle of a linked list.</td>
<td>$\Theta(N)$</td>
<td>All cases the same.</td>
</tr>
<tr>
<td>Add an element to the beginning of a linked list.</td>
<td>$\Theta(1)$</td>
<td>All cases the same.</td>
</tr>
<tr>
<td>Algorithm and Data Input</td>
<td>$\Theta(......)$</td>
<td></td>
</tr>
<tr>
<td>Sort an already-sorted list using Selection Sort</td>
<td>$\Theta(N^2)$</td>
<td></td>
</tr>
<tr>
<td>Sort an already-sorted list using Insertion Sort</td>
<td>$\Theta(N)$</td>
<td></td>
</tr>
<tr>
<td>Sort an already-sorted list using Mergesort</td>
<td>$\Theta(N \log N)$</td>
<td></td>
</tr>
<tr>
<td>Sort an already-sorted list using Quicksort</td>
<td>$\Theta(N^2)$</td>
<td></td>
</tr>
</tbody>
</table>