MIDTERM (with SOLUTIONS)

Problem 1 TRUE or FALSE [5+5 points]

True/False answers must be briefly justified to get any credit.

(a) We can find the second largest key in a max-heap in constant time.

ANSWER: True: the second largest is either $A[1]$ or $A[2]$.

(b) We can delete the second largest key in a max-heap in constant time (and maintain the max-heap after deletion).

ANSWER: False: deleting a single node will cause a non-constant number of nodes to be reshuffled. (Essentially, we need to do siftdown and this takes logarithmic time.)

Problem 2 ASYMPTOTIC ANALYSIS [10+15+15 points]

(a) Show that

$$f(x) = O(g(x))$$

where $f(x) = 3x^4 + 8x^2 - x$ and $g(x) = x^4$. Note: you should show two implicit constants C and x_0.

ANSWER: We must find $C > 0$ and x_0 such that

$$f(x) = 3x^4 + 8x^2 - x \leq C \cdot g(x) = 4x^4 \tag{1}$$

whenever $x \geq x_0$. If $x_0 \geq 0$, then the inequality (1) follows from $3x^4 + 8x^2 \leq 4x^4$. But $3x^4 + 8x^2 \leq 4x^4$ is equivalent to $8x^2 \leq x^4$ which is equivalent to $8 \leq x^2$. Hence if $x_0 = \sqrt{8}$ then the claim is true whenever $x \geq x_0$. As required above, we have chosen $x_0 \geq 0$. So we can choose $C = 4$ and $x_0 = \sqrt{8}$.

(b) Order the following functions according to increasing asymptotic order of growth:

(i) x (ii) 3^{-x} (iii) $x^{1/(\lg x)}$ (iv) $\lg^2 x$
(v) $\sqrt{x} + \lg^3 x$ (vi) $x!$ (vii) $x^3 \lg x - \lg^4 x$ (viii) 2^x

HINT: To understand the expression (iii), take its logarithm!

ANSWER:

After taking log (base 2), (iii) becomes 1. Hence (iii) must be equal to 2.

(ii) 3^{-x} (iii) $2 = x^{1/(\lg x)}$ (iv) $\lg^2 x$ (v) $\sqrt{x} + \lg^3 x$
(i) x (vii) $x^3 \lg x - \lg^4 x$ (viii) 2^x (vi) $x!$

(c) Let

$$T(x) = \begin{cases} 2T(x/2) + x^2 & \text{if } x > 1 \\ 0 & \text{else.} \end{cases}$$

Show that $T(x) = \Theta(x^2)$. HINT: Which half of this is trivial? For the other half, you can use the rote method or real induction.

ANSWER: The trivial part is $T(x) = \Omega(x^2)$. This is because $T(x) = 2T(x/2) + x^2 \geq$

x^2. So we must prove $T(x) = O(x^2)$. Real induction is the simplest. The proof: we claim the following predicate $P(x)$ holds:

$$P(x) : \text{For all } x \geq 1,\ T(x) \leq 2x^2.$$

Choose $x_0 = 2$ and $\delta = 1$. If $x < 1$, $P(x)$ holds trivially. If $1 < x \leq x_0$, we may verify that $P(x)$ holds because

$$T(x) = 2T(x/2) + x^2 = x^2.$$

It remains to prove that for $x > x_0$, $P(x)$ holds. By induction hypothesis, we may assume that for all $y \leq x - \delta$, $P(y)$ holds. Therefore,

$$\begin{aligned} T(x) &= 2T\left(\frac{x}{2}\right) + x^2 \\ &\leq 2[2\left(\frac{x}{2}\right)^2] + x^2 \\ &= 2x^2. \end{aligned}$$

You can also use the rote method:

$$\begin{aligned} T(x) &= 2T\left(\frac{x}{2}\right) + x^2 \\ &= 2[2T\left(\frac{x}{4}\right) + \left(\frac{x}{2}\right)] + x^2 \qquad \text{(EXPAND)} \\ &= 2^2T\left(\frac{x}{2^2}\right) + x^2[1 + 2^{-1}] \\ &= \cdots \\ &= 2^iT\left(\frac{x}{2^i}\right) + x^2\sum_{j=0}^{i-1} 2^{-j}, \qquad \text{(GUESS)} \\ &= 2^i[2T\left(\frac{x}{2^{i+1}}\right) + \left(\frac{x}{2^i}\right)^2] + x^2\sum_{j=0}^{i-1} 2^{-j}, \qquad \text{(VERIFY)} \\ &= 2^{i+1}T\left(\frac{x}{2^{i+1}}\right) + x^2\sum_{j=0}^{i} 2^{-j}, \qquad \text{(VERIFIED!)} \\ &= x^2\sum_{j=0}^{i} 2^{-j}, \qquad \text{(STOP at } i = \lceil \lg x \rceil) \\ &< 2x^2 \qquad (\text{since } \textstyle\sum_{j=0}^{\infty} 2^{-j} = 2.) \end{aligned}$$

Problem 3 BINARY TREES [15 points]

Let T be an arbitrary binary tree with n nodes. We want you to use strong induction on n to show that T has height at least $\lfloor \lg n \rfloor$. Note that if $n = 1$, the height is 0. HINT: the height is minimized if the left and right subtrees of T have almost equal size. Consider two cases depending on wheter n is odd or even.

ANSWER: Result is true for $n = 1$: $\lfloor \lg 1 \rfloor = 0$. For $n > 1$, it is easy to see that the maximum height of the two subtrees of T is at least $\lfloor \lg \lceil (n-1)/2 \rceil \rfloor$. So, we need to prove

$$\lfloor \lg n \rfloor = 1 + \lfloor \lg \lceil (n-1)/2 \rceil \rfloor .$$

Consider 2 cases. EVEN CASE: Say $n = 2m$. Then the left hand side (LHS) is equal to $\lfloor \lg 2m \rfloor = 1 + \lfloor \lg m \rfloor$. The right hand side (RHS) is also easily seen to be $1 + \lfloor \lg m \rfloor$. ODD CASE: Say $n = 2m+1$. Then the LHS is equal to $\lfloor \lg(2m+1) \rfloor = \lfloor \lg(2m) \rfloor = 1 + \lfloor \lg m \rfloor$. We check that the RHS is also $1 + \lfloor \lg m \rfloor$.

Problem 4 BINARY SEARCH TREES [8 + 5 + 5 + 7 points]

Suppose you insert the following sequence of keys (in this order) into an initially empty binary search tree:

$$10, 8, 4, 6, 19, 13, 18, 14, 16.$$

(a) Draw the binary search tree after each insertion. Let T be the final binary tree. ANSWER:

```
          10
         /  \
       8      19
      /      /
    4      13
     \       \
      6       18
             /
           14
             \
              16
```

(b) Draw the result of deleting key 13 from T.

```
          10
         /  \
       8      19
      /      /
    4      18
     \     /
      6   14
            \
             16
```

COMMENT: some students replaced 13 by the successor of 13, which is 14, and then they delete 14. This is not the correct algorithm.

(c) Specify the ranges of real values such that if you insert into T a key from these

ranges, you increase the height of T.
ANSWER: Inserting any key x in the ranges $14 < x < 16$ or $16 < x < 18$ will increase the height of T.
(d) List the keys from a post-order traversal of T.
ANSWER: $6, 4, 8, 16, 14, 18, 13, 19, 10$

PROBLEM 5 JAVA [25 points]
Your Java program below should compile without error. If you forget any particular Java construction, do not let this deter you from writing the rest of the program. E.g., if you need to compare two strings but forgot the method for doing this, just make a reasonable assumption and insert a comment to tell us about your assumption. Do not forget casting between `nodeInt` and `myNode`.

Assume (similar to homework 3) that singly-linked lists are represented by the interface `nodeInt` which in turn is implemented by the class `myNode`:

```
class myNode implements nodeInt {
    private String item;
    private myNode next;
    public String getItem(){...}
    public nodeInt getNext(){...}
    ...
}
```

Write a method for this class whose header (or "stub") has the form

`public myNode suffix(String s).`

Let L be the linked list whose head node is `N`. Basically, `N.suffix(s)` splits up the list L into two sublists at the node containing s. More precisely, if `N1` is the first node in L such that `N1.item = s`, then `N.suffix(s)` returns the node `N1.getNext()`, and the list L is modified so that the last node is `N1`. If string `s` does not occur in L, the `N.suffix(s)` returns `null` and L is unchanged. E.g., if the N is the head of the list `("aa", "ab", "ba", "bb", "bc")` of strings, then `N.suffix("ba")` returns the header to the list `("bb", "bc")` while L is transformed to the list `("aa", "ab", "ba")`. But `N.suffix("bc")` or `N.suffix("dd")` both returns `null` without modifying the list L.

ANSWER:

```
 \tt
public myNode suffix(String s)\{
  myNode currN = this; // Note initialization of currN
  myNode N1; // Node when string s is found

  while (currN != null) \{
    if (currN.getItem().compareTo(s) == 0) \{  // Note "compareTo"
```

```
N1 = (myNode)currN.getNext(); // head of new list
currN.setNext(null); // truncate original list
return N1;
    \} else
    currN = (myNode)currN.getNext();
  \}

  return null; // This is when N1 == null.
\}
```

COMMENTS: Some students compare strings using `currN.getItem() == s`. Why is this wrong? Note that there is no need to call `new` to initialize `myNode currN`, `N1`, etc. Why? Note that we assign to N1 before truncating the list. Some student "return N1" and then try to truncate the list. Why is this wrong?