COS 226	Algorithms and Data Structures	Fall 2006

Midterm

This test has 8 questions worth a total of 50 points. You have 80 minutes. The exam is closed book, except that you are allowed to use a one page cheatsheet. No calculators or other electronic devices are permitted. Give your answers and show your work in the space provided. **Write out and sign the Honor Code pledge before turning in the test.**

"I pledge my honor that I have not violated the Honor Code during this examination."

Problem	Score
1	
2	
3	
4	
Sub 1	

Problem	Score
5	
6	
7	
8	
Sub 2	

Total	

Name:

Login ID:

Precept:

1	12:30	Janet
3	3:30	Wolfgang

1. **8 sorting algorithms. (8 points)**

   The column on the left is the original input of strings to be sorted. The columns to the right are the contents at some intermediate step during one of the 8 sorting algorithms listed below. Match up each algorithm by writing its number under the corresponding column. Use each number exactly once.

```
Data |   Leaf    Code    Code    Case    Type    Next    Data    Code    Case
Type |   Scan    Cost    Data    Code    Trie    Edge    Edge    Data    Code
Hash |   Heap    Case    Find    Cost    Tree    Hash    Hash    Find    Cost
Code |   Swap    Data    Hash    Data    Time    Code    Code    Hash    Data
Heap |   Exch    Exch    Heap    Edge    Loop    Heap    Heap    Heap    Edge
Sort |   Code    Edge    Link    Exch    Swim    Lifo    Lifo    Leaf    Exch
Link |   Node    Find    List    Fifo    Temp    Link    Link    Left    Fifo
List |   Tree    Fifo    Loop    Find    Skip    List    List    Link    Find
Push |   Fifo    Hash    Push    Hash    Push    Push    Push    List    Hash
Loop |   Lifo    Heap    Root    Heap    Heap    Loop    Loop    Loop    Heap
Find |   Left    Join    Sort    Join    Join    Find    Find    Node    Join
Root |   Edge    Link    Type    Leaf    Sort    Root    Root    Null    Leaf
Leaf |   Trie    List    Leaf    Root    Scan    Leaf    Leaf    Path    Left
Tree |   Swim    Loop    Tree    Tree    Swap    Fifo    Fifo    Push    Less
Null |   Join    Leaf    Null    Null    Null    Null    Null    Root    Lifo
Path |   Skip    Left    Path    Path    Path    Path    Path    Sort    Link
Node |   Null    Less    Node    Node    Node    Node    Node    Tree    List
Left |   Time    Lifo    Left    Left    Left    Left    Left    Type    Loop
Less |   Temp    Null    Less    Less    Less    Less    Less    Case    Next
Cost |   Find    Node    Cost    Push    Cost    Cost    Cost    Cost    Node
Case |   Link    Next    Case    Type    Case    Case    Case    Edge    Null
Join |   Sink    Push    Join    List    Find    Join    Join    Exch    Path
Exch |   Loop    Path    Exch    Sort    Exch    Exch    Exch    Fifo    Push
Sink |   Root    Root    Sink    Sink    Sink    Data    Sink    Join    Root
Swim |   Type    Sort    Swim    Swim    Root    Scan    Scan    Less    Scan
Next |   Sort    Sink    Next    Next    Next    Swap    Next    Lifo    Sink
Scan |   Case    Swim    Scan    Scan    Leaf    Skip    Skip    Next    Skip
Swap |   Hash    Scan    Swap    Swap    Hash    Sort    Swap    Scan    Sort
Temp |   Push    Swap    Temp    Temp    Link    Sink    Temp    Sink    Swap
Fifo |   Less    Skip    Fifo    Link    Fifo    Swim    Tree    Skip    Swim
Lifo |   List    Type    Lifo    Lifo    Lifo    Time    Sort    Swap    Temp
Trie |   Cost    Tree    Trie    Trie    List    Tree    Trie    Swim    Time
Edge |   Data    Temp    Edge    Loop    Edge    Temp    Type    Temp    Tree
Time |   Path    Trie    Time    Time    Data    Type    Time    Time    Trie
Skip |   Next    Time    Skip    Skip    Code    Trie    Swim    Trie    Type
----     ----    ----    ----    ----    ----    ----    ----    ----    ----
  0
```

(0) Original input

(1) 3-way radix quicksort

(2) Heap sort

(3) Insertion sort

(4) LSD radix sort

(5) Mergesort

(6) MSD radix sort

(7) Quicksort

(8) Selection sort

(9) All of them

2. **Algorithm Properties. (6 points)**

   Match up each *worst-case* quantity on the left with the best matching order-of-growth term on the right. You may use a letter more than once.

___ Height of a binary heap with N keys	A. 1
___ Height of a BST with N keys	B. $\log N$
___ Number of comparisons to quicksort N equal keys using our standard version of quicksort	C. N
___ Number of comparisons to quicksort N equal keys using 3-way quicksort	D. $N \log N$
___ Time to iterate over the keys in a BST using inorder traversal	E. N^2
___ Number of equality tests to insert N keys into an empty linear probing hash table of size $2N$.	F. 2^N

3. **Sorting a linked list. (6 points)**

   Suppose that you wish to sort a singly *linked list* of N `Comparable` items. Which algorithm would you choose and why? For your algorithm, describe its (i) memory usage, beyond the space required to represent the linked list, (ii) average asymptotic number of compares, and (iii) whether or not the algorithm is stable.

Algorithm	Extra memory	Running time	Stability

4. **Comparable interface. (4 points)**

What is *broken* with the following implementation of the Java `Comparable` interface?

```
public class Temperature implements Comparable<Temperature> {
    private double degrees;   // Kelvin

    public Temperature(double degrees) {
        this.degrees = degrees;
    }
    public int compareTo(Temperature y) {
        double EPSILON = 0.01;
        if      (degrees < y.degrees - EPSILON)  return -1;
        else if (degrees > y.degrees + EPSILON)  return +1;
        else                                     return  0;
    }
}
```

5. **Java API. (4 points)**

You have been hired to design a new Java library with the following API.

```
public class OrderStatistic<Item extends Comparable>
   public boolean isEmpty()        // is the data structure empty?
   public int size()               // return the number of items N
   public void insert(Item item)   // insert an item
   public Item select(int k)       // return the kth largest item for 1 <= k <= N
```

Your manager requires that all operations take *constant* time in the worst-case.
Describe why you won't succeed.

6. **Binary heaps. (6 points)**

   Consider the following binary heap (i.e., the array-representation of a heap-ordered complete binary tree).

0	1	2	3	4	5	6	7	8	9	10	11	12	13
-	X	W	J	V	U	D	H	S	P	Q	R	C	-

   (a) *Draw* the corresponding binary tree.

   (b) *Insert* the key M. Give the resulting binary heap, circling those values that changed.

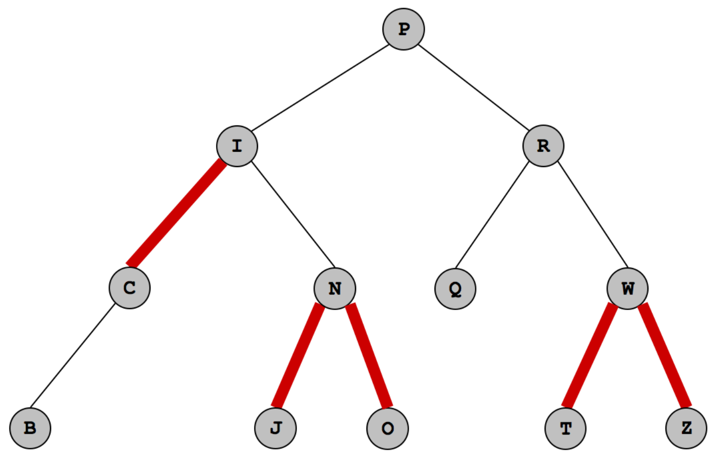
0	1	2	3	4	5	6	7	8	9	10	11	12	13
-													

   (c) *Delete* the maximum key from the *original* binary heap. Give the resulting binary heap, circling those values that changed.

0	1	2	3	4	5	6	7	8	9	10	11	12	13
-													

7. **Red-black trees. (6 points)**

Consider the following red-black tree. (As usual, the thick edges represent red links.)



Add the key S; then add the key D. Draw the final red-black tree.

8. **Two-sum. (10 points)**

TwoSum. Given an array of N 64-bit `long` integers, find two integers x and y such that $x + y = 0$. (For simplicity, assume none of the integers is 0 or -2^{63}.)

(a) Describe a efficient algorithm for TwoSum in the box below. Your algorithm should run in linear time on average (for full credit) or linearithmic time (for partial credit). Your answer will be graded on correctness, clarity, and conciseness.

(b) Circle the *average-case* running time of your algorithm.

$\log N$ N $N \log N$ N^2 2^N

(c) Circle the *worst-case* running time of your algorithm.

$\log N$ N $N \log N$ N^2 2^N