NAME:

login ID:
precept:

COS 226 Midterm Exam Solutions, Fall 2007

This test is 10 questions, weighted as indicated. 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. *Put your name, login ID, and precept number on this page* (*now*), and write out and sign the Honor Code pledge before turning in the test. You have 80 minutes to complete the test.

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

1	/5
2	/10
3	/5
4	/10
5	/10
6	/10
7	/10
8	/5
9	/25
10	/10
TOTAL	/100

October 25, 2007

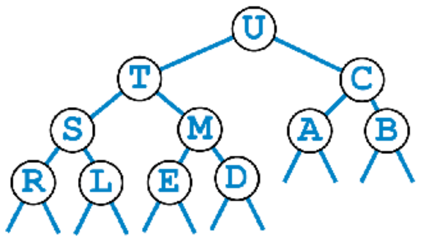
1. **Partitioning** (5 points). Give the result of partitioning the array with standard Quicksort partitioning (taking the rightmost N as the partitioning element).

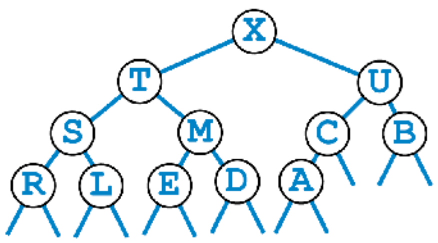
```
P A R T I T I O N I N G Q U E S T I O N
I                                 P
    E                       R
      G               T
          T         N
              I   O
                N                     N

I A E G I N I I N O T T Q U R S T P O N
```

2. **Heap representation** (10 points). Draw the complete heap-ordered tree that corresponds to the following array representation of a max-heap, then draw the result of inserting X into the heap, then give the contents of the heap-ordered array corresponding to your result.

initial contents

i	0	1	2	3	4	5	6	7	8	9	10	11
a[i]	-	U	T	C	S	M	A	B	R	L	E	D





result of inserting X *(circle all elements that changed)*

i	0	1	2	3	4	5	6	7	8	9	10	11	12
a[i]	-	X*	T	U*	S	M	C*	B	R	B	L	D	A*

1. **Mystery code** (5 points). Circle the choice that describes a reasonable use of the following code:

```
Comparable v = a[lo];
for (int j = lo; j <= hi; j++)
    if      (less(v, a[j])) exch(a, j--, hi--);
    else if (less(a[j], v)) exch(a, lo++, j);
```

A. To heap-order an array

B. For merging in a recursive mergesort

C. To build the array corresponding to a 2-3-4 tree

D. For partitioning in a recursive quicksort

E. As one pass in a shellsort

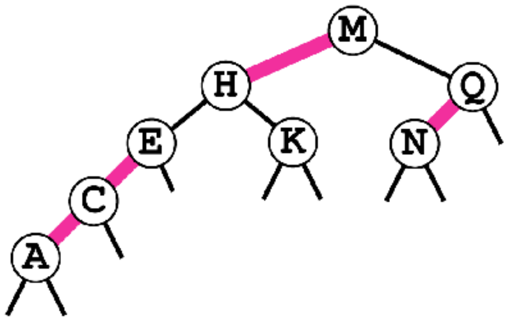
2. **Tree height** (10 points). Write *O(1)*, *O(log* N)*, *O(log N)*, or *O(N)* in the blank following each of the following tree structures to best describe the guaranteed difference in the distances to the root for any two nodes in the tree (using the best probabilistic or worst-case guarantee).

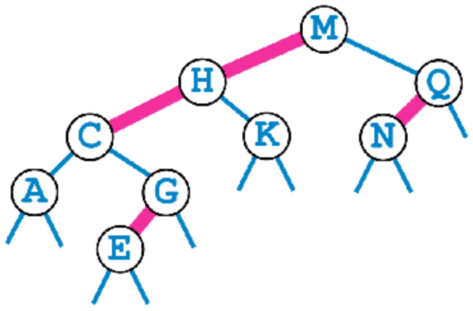
A. BST	`O(N)`
B. Heap-ordered complete tree	`O(log N)`
C. Left-leaning red-black tree	`O(log N)`
D. BST with root insertion	`O(N)`
E. randomized BST	`O(log N)`
F. quick-union with path compression	`O(N)`
G. weighted quick-union	`O(log N)`

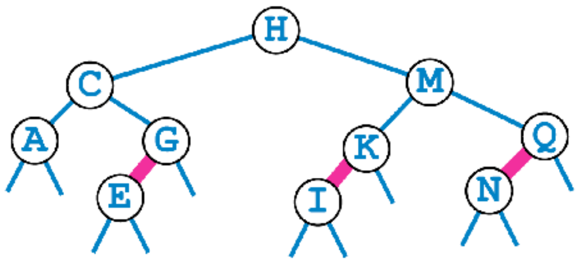
3. **Data moves for sorts** (10 points). Write *linear*, *linearithmic*, or *quadratic* in the blank following each of the following sorting algorithms to best describe the number of times that they assign a value to an array entry, for a randomly ordered array of distinct values.

A. Insertion sort	`quadratic`
B. Mergesort	`linearithmic`
C. Quicksort	`linearithmic`
D. Heapsort	`linearithmic`
E. Selection sort	`linear`

4. **LLRB insertion** (10 points). Draw the result of inserting `G` into the following left-leaning red-black tree, then draw the result of inserting `I` into the tree that you have drawn.







5. **ST implementations** (10 points). The following is a list of possible reasons for choosing one of the Java ST implementations given in lecture over another. In the blanks provided, first list the ones that might reasonably justify using red-black trees rather than hash tables, then list the ones that might reasonably justify using hash tables over red-black trees. You need not use all the choices (do not list a choice if there are reasonable arguments on both sides).

   A. Easier to use properly for built-in key types (such as `String` and `Integer`)

   B. Easier to use properly for user-defined key types

   C. Extends to handle useful operations for ordered keys

   D. Uses less space

   E. Better worst-case performance guarantee

   F. Faster for `int` keys

   G. Better Java system support

   Reasons to use red-black trees: B C E

   Reasons to use hash tables: D F G

6. **Heap positions** (5 points). Suppose that an array `a[]` is a max-heap that contains the distinct integer keys `1, 2, 3, ... N` with `N` larger than 2. The key `N` must be in `a[1]` and the key `N-1` must must be in `a[2]` or `a[3]`. Give all possible positions for the key `N-2`.

   `a[2]   a[3]   also a[4] a[5] a[6] a[7] if they exist`

   Answer the question for the key `2`.

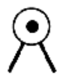
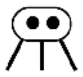
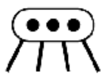
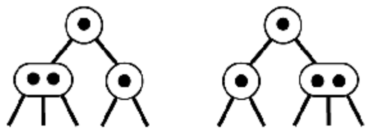
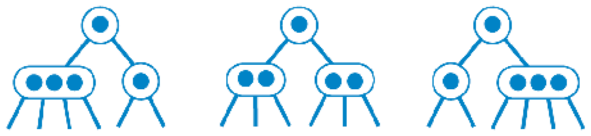
   `a[ceil(N/2)] through a[N]`

7. **7 sorting algorithms** (25 points). The leftmost column is the original input of strings to be sorted, and the rightmost column is the sorted result. The other columns are the contents at some intermediate step during one of the 7 sorting algorithms listed below. Match up each algorithm by writing its letter under the corresponding column. Use each letter exactly once.

fuzz	zoom	doze	benz	cozy	cozy	benz	cozy	benz
cozy	zest	cozy	cozy	czar	czar	buzz	fuzz	buzz
zinc	zone	czar	zinc	doze	fuzz	cozy	quiz	cozy
quiz	ritz	benz	quiz	fuzz	hazy	cruz	zinc	cruz
zero	zero	faze	zero	gaze	laze	czar	hazy	czar
suez	suez	cruz	suez	hazy	maze	daze	suez	daze
zone	zing	haze	zone	jazz	ouzo	doze	zero	doze
hazy	quiz	buzz	hazy	laze	quiz	faze	zone	faze
maze	maze	fuzz	maze	maze	suez	fuzz	czar	fuzz
ouzo	ouzo	gaze	ouzo	ouzo	zero	gaze	laze	gaze
czar	zeal	ritz	czar	quiz	zinc	haze	maze	haze
laze	raze	daze	laze	suez	zone	hazy	ouzo	hazy
doze	lazy	maze	doze	zero	doze	zone	doze	jazz
gaze	zeta	lazy	gaze	zinc	gaze	ouzo	gaze	laze
zing	zinc	whiz	zing	zing	zing	zing	jazz	lazy
jazz	jazz	hazy	jazz	zone	jazz	jazz	zing	maze
zoom	hazy	ooze	zoom	benz	zoom	zoom	buzz	ooze
cruz	cruz	ouzo	cruz	buzz	cruz	quiz	cruz	ouzo
ritz	cozy	zeal	ritz	cruz	ritz	ritz	ritz	quiz
buzz	buzz	jazz	buzz	daze	buzz	zinc	zoom	raze
faze	faze	zero	faze	faze	faze	laze	faze	ritz
zest	czar	size	zest	haze	zest	zest	raze	size
zeal	fuzz	zinc	zeal	lazy	zeal	zeal	zeal	suez
raze	laze	laze	raze	ooze	raze	raze	zest	whiz
ooze	ooze	zeta	ooze	raze	ooze	ooze	daze	zeal
lazy	doze	suez	lazy	ritz	lazy	lazy	haze	zero
haze	haze	zing	haze	size	haze	zero	lazy	zest
daze	daze	quiz	daze	whiz	daze	suez	ooze	zeta
zeta	gaze	zoom	zeta	zeal	zeta	zeta	benz	zinc
size	size	zest	size	zest	size	size	size	zing
whiz	whiz	zone	whiz	zeta	whiz	whiz	whiz	zone
benz	benz	raze	fuzz	zoom	benz	maze	zeta	zoom
	B	F	D	G	C	E	A	

A. Bottom-up mergesort
B. Heapsort
C. Insertion sort
D. Quicksort (with no random shuffle)
E. Selection sort
F. Shellsort
G. Top-down mergesort

8. **2-3-4 trees** (10 points). The table below lists all possible 2-3-4 tree shapes that could result from inserting *N* distinct keys into an initially empty tree using top-down insertion, for *N* between 1 and 6. The left column is the number of keys, the next column is the number of possible trees with that many keys, all of which are drawn on the right (with dots indicating the key values). Complete the two bottom rows of the table (draw the three trees with 5 keys and enter the count and draw the trees with 6 keys).

1	1	
2	1	
3	1	
4	2	
5	3	
6	5