NAME:

login ID: **Precept (circle one):** `P01 P01A P01B P02 P02A P03`

COS 226 Midterm Exam, Spring 2011

This test is 9 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	/12
2	/10
3	/10
4	/10
5	/14
6	/10
7	/10
8	/14
9	/10
TOTAL	/100

March 7, 2011

1. **Partitioning** (12 points).

A. (4 points) Fill in the diagram below with the result of partitioning the array with *standard quicksort* partitioning (taking the `E` at the left as the partitioning item). Also give the number of exchanges.

```
E   V   E   R   Y   E   Q   U   A   L   K   E   Y   S   T   O   P   S   I   T
```


Number of exchanges ___________

B. (3 points) Why does standard quicksort partitioning stop the partitioning scans on keys equal to the partitioning item? Circle your answer.

a. It is a lazy algorithm.

b. To avoid exchanging them.

c. To avoid quadratic running time.

d. To make the algorithm stable.

e. None of the above.

C. (5 points) Fill in the diagram below with the result of partitioning the array with *3-way quicksort partitioning* (again taking the `E` at the left as the partitioning item). Also give the number of exchanges.

```
E  V  E  R  Y  E  Q  U  A  L  K  E  Y  S  T  O  P  S  I  T
```

□□□□□□□□□□□□□□□□□□□□

Number of exchanges ___________

2. **Estimating order of growth** (10 points).

A. (5 points) Suppose that you observe that a program takes 30 seconds to complete on inputs of size 600 and 4.5 minutes to complete on inputs of size 1800. Develop a reasonable hypothesis for the order of growth of the running time. *Hint*: Use a tripling hypothesis.

order of growth: ____________

B. (5 points) Suppose that you observe that a program takes x seconds to complete on inputs of size N and y seconds to complete on inputs of size $10N$. Under the hypothesis that the order of growth of the running time is N to a power b, give a formula for b.

$b =$ ____________

3. **Sorting algorithms** (10 points). Give the order of growth of the running times of the following sorting algorithms for arrays that are already in order and for arrays that are in reverse order. For quicksort, note that it randomizes, so give the expected order of growth. Write *constant, N, N log N,* N^2, or *?* in each blank. Fill in all blanks and use exactly one question mark (in other words, there is one answer that we think is an open research question).

sorted order reverse order

a. Quicksort ______________ ________________

b. Heapsort ______________ ________________

c. Insertion sort ______________ ________________

d. Selection sort ______________ ________________

e. Shellsort ______________ ________________
(with increments 1, 4, 13, 40, ...)

4. **Tree height** (10 points). Write *constant, log N, log*N, or N* in the blank following each of the following tree structures to best describe the order of growth of the height of an *N*-node tree in the best case and in the worst case.

best case worst case

a. BST ___________ __________

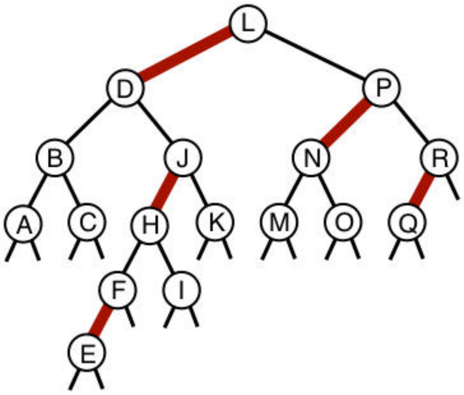
b. Heap-ordered complete tree ___________ __________

c. Left-leaning red-black tree ___________ __________

d. quick-union with path compression ___________ __________

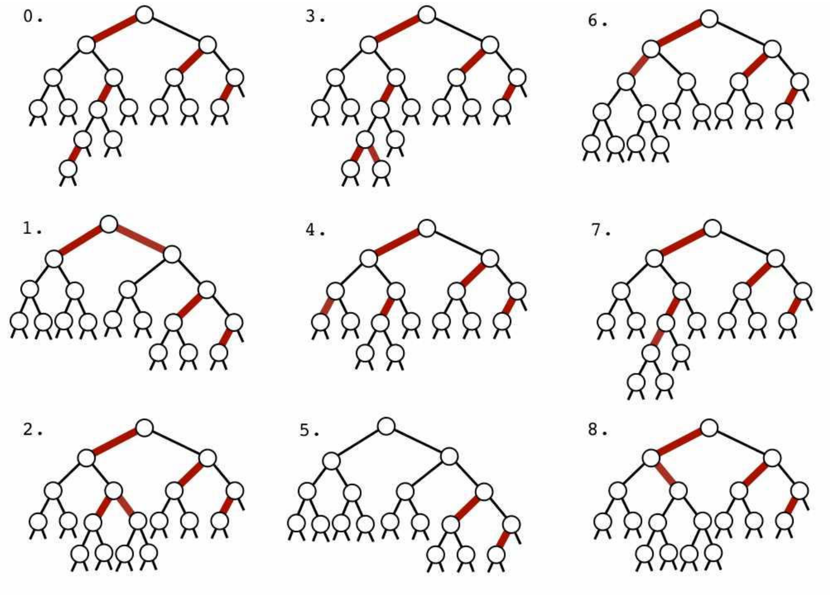
e. weighted quick-union ___________ __________

5. **LLRB insertion** (14 points). The following diagram shows a left-leaning red-black BST Thick lines are red links.



a. (5 points) Draw the left-leaning red-black BST that results after S is inserted.

b. (2 points) Which of the trees drawn below are legal left-leaning red-black BSTs? In the blank below, write the numbers corresponding to all legal LLRB BSTs. Do not include trees that might arise at *intermediate* stages of an insertion.

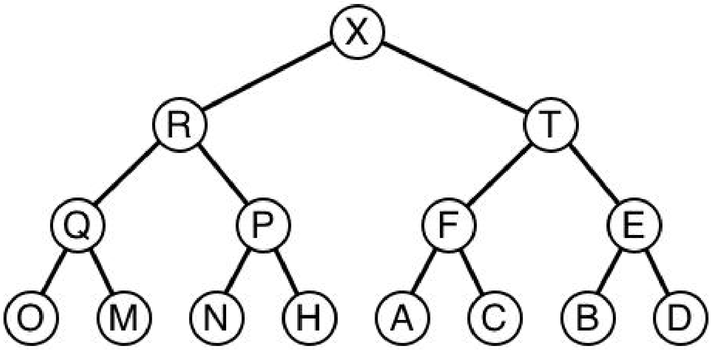


Legal LLRB trees ____________________________

c. (7 points) Seven of the trees drawn above are intermediate stages when a new key is inserted into tree 0. In the blanks below, list the numbers corresponding to these trees (not including tree 0) in the order in which they occur during the insertion process.

______ ______ ______ ______ ______ ______ ______

6. **Heap operations** (10 points). Consider the following max-heap:



a. (5 points) Draw the result of inserting `S`.

b. (5 points) Draw the result of deleting the maximum from the **original** max-heap shown above (**before** `S` has been inserted).

7. **Linear probing** (10 points). A programmer building a symbol-table with single-character keys uses a bad hash function where keys in the first half of the alphabet (A through M) hash to the value 6 and keys in the second half of the alphabet (N through Z) hash to the value 9.

   A. (6 points) Fill in the diagram below to give the result of inserting the keys

   `B A D F U N C T I O N`

   into an initially empty table of size 11 with this hash function.

0	1	2	3	4	5	6	7	8	9	10

   B. (4 points) Suppose that the same programmer has the same problem with *N* keys (half the keys hash to the value 6 and the other half hash to the value 9). Answer the following questions about the order of growth of the running time for certain operations by filling in the blanks below with one of the words *constant, linear, linearithmic, or quadratic*. For linear probing assume the array size to be 2*N*; for separate chaining, assume the array size to be *N*/5.

   a. Average time for a search hit with separate chaining when searching for a random key ______________

   b. Average time for a search hit with linear probing when searching for a random key ______________

   c. Total time to build the table with linear probing ______________

   d. Total time to build the table with separate chaining ______________

8. **7 sorting algorithms** (14 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.

```
that  from  been  good  your  been  from  have  been
with  good  from  have  with  that  have  that  from
have  have  good  been  will  good  know  that  good
this  that  have  know  this  have  that  this  have
that  been  know  from  they  from  that  will  know
will  that  that  that  want  that  that  will  that
will  know  that  that  will  know  they  with  that
your  that  that  that  that  that  this  your  that
from  your  that  that  from  that  will  from  that
they  they  they  they  that  they  will  know  they
know  will  this  your  know  will  with  that  this
that  will  want  want  that  this  your  they  want
want  want  will  will  have  want  want  been  will
been  that  your  will  been  will  been  good  will
that  this  with  this  that  with  that  that  with
good  with  will  with  good  your  good  want  your
```

input ___ ___ ___ ___ ___ ___ ___ result

a. 3-way quicksort (with no random shuffle)
b. Shellsort (13-4-1 increments)
c. Insertion sort
d. Quicksort (with no random shuffle)
e. Selection sort
f. Top-down mergesort
g. Heapsort

9. **Maximum difference** (10 points). Consider the following code fragment which computes the *maxdiff* function for an array: the maximum value of the difference between an entry and an entry to its left. For example, the maxdiff of `{3, 1, 4, 1, 5, 9, 2}` is `8` because the difference between the 9 and the 1s to its left is 8 and no other difference between an entry and another entry to its left is greater. As another example the maxdiff of `{9, 4, 6, 8, 2, 2}` is `4`.

```
public static int slow(int[] a)
{
   int max = a[1] - a[0];
   for (int j = 2; j < a.length; j++)
      for (int i = 0; i < j; i++)
         if (a[j] - a[i] > max) max = a[j] - a[i];
   return max;
}
```

An enterprising COS226 student figured out a faster way to solve the problem, with the following code:

```
public static int fast(int[] a)
{
   int max = a[1] - a[0];
   int min = a[0];
   for (int k = 1; k < a.length; k++)
   {
      // First missing line of code.
      // Second missing line of code.
   }
   return max;
}
```

A. (8 points) In the space below, write the *two lines* of code missing from `fast()`. A "line of code" is a statement ending with a semicolon.

____

____

B. (2 points) What is the order of growth of the running time of the two methods? Write one of the following words in each of the blanks below: *constant, linear, linearithmic, quadratic, cubic.*

`slow()` ________________

fast() ________________