COS 226	Algorithms and Data Structures	Spring 2012

Midterm

This test has 9 questions worth a total of 60 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
0	
1	
2	
3	
4	
Sub 1	

Problem	Score
5	
6	
7	
8	
Sub 2	

Total	

Name:

Login ID:

Precept:

P01	Th 12:30	Diego Botero
P01A	Th 12:30	David Shue
P01B	Th 12:30	Joey Dodds
P02	Th 1:30	Josh Hug
P03	Th 3:30	Josh Hug
P04	F 11	Joey Dodds
P04A	F 11	Jacopo Cesareo

0. **Miscellaneous. (1 point)**

   In the space provided on the front of the exam, write your name and Princeton NetID; circle your precept number; and write out and sign the honor code.

1. **Analysis of algorithms. (5 points)**

   Consider the following code fragment, where `a[]` is an array of N `Point2D` objects.

```
int min = N;
for (int i = 0; i < N; i++) {
   Selection.sort(a, a[i].BY_POLAR_ORDER);
   for (int j = 0; j < N; j++) {
        for (int k = j+1; k < N; k++) {
             if (a[j].distanceTo(a[k]) <= 1.0) {
                  min = Math.min(min, k - j);
             }
        }
   }
}
```

   (a) Suppose that the code fragment takes 30 seconds when $N = 2,000$. Estimate the running time (in seconds) as a function of the input size N. Use tilde notation to simply your answer.

   (b) Suppose that you replace the call to `Selection.sort()` with a call to `Merge.sort()`. What is the order of growth of the running time of the modified 11-line code fragment as a function of N? Circle the best answer.

   1 $\quad N$ $\quad N \log N$ $\quad N^2$ $\quad N^2 \log N$ $\quad N^3$ $\quad 2^N$

2. **Data structure and algorithm properties. (8 points)**

(a) Match up each quantity on the left with the best matching quantity on the right. You may use a letter more than once or not at all.

___ *Min* height of a binary heap with N keys.	A. ~ 1
___ *Max* height of a binary heap with N keys.	B. $\sim \frac{1}{2} \lg N$
___ *Min* height of a 2-3 tree with N keys.	C. $\sim \log_3 N$
___ *Max* height of a 2-3 tree with N keys.	D. $\sim \ln N$
___ *Min* height of left-leaning red-black BST with N keys.	E. $\sim \lg N$
___ *Max* height of left-leaning red-black BST with N keys.	F. $\sim 2 \lg N$
___ *Min* height of a weighted quick union tree with N items.	G. $\sim 2 \ln N$
___ *Max* height of a weighted quick union tree with N items.	H. $\sim N$

(b) A sorting algorithm is *parsimonious* if no pair of items is compared more than once. Circle the following sorting algorithms (as implemented in lecture and the textbook) if they are parsimonious; cross them out if they are not parsimonious.

insertion sort   selection sort   top-down mergesort   heapsort

3. **Data structures. (8 points)**

The Java library `StringBuilder` represents a mutable sequence of characters. Suppose that it is implemented using a resizing array (doubling when full and halving when one-quarter full), maintaining one instance variable `N` to count the number of characters in the sequence and another instance variable `a[]` to hold the sequence of characters.

```
public class StringBuilder {
   private int N;       // number of characters in the sequence
   private char[] a;    // the character sequence a[0], a[1], ..., a[N-1]
   ...
}
```

(a) Using the 64-bit memory cost model from the textbook, how much memory (in bytes) does a `StringBuilder` object use to store a sequence of N characters? Simplify your answer using tilde notation.

- Best case:

- Worst case:

(b) What is the order of growth of the *amortized running time* of each of operation below? Write down the best answer in the space provided, using one of the following functions.

1 $\log N$ $\sqrt{N}$ N $N \log N$ N^2

operation	*description*	*running time*
`charAt(int i)`	*return the ith character in sequence*	
`deleteCharAt(int i)`	*delete the ith character in the sequence*	
`append(char c)`	*append c to the end of the sequence*	
`set(int i, char c)`	*replace the ith character with c*	

4. **8 sorting and shuffling algorithms. (8 points)**

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

lynx	bass	lion	bass	bass	bass	bass	gnat	wren	bass
bass	bear	frog	bear	bear	bear	bear	bass	worm	bear
bear	crab	mole	clam	crab	clam	clam	bear	oryx	clam
crab	lion	hawk	crab	lynx	crab	crab	crab	swan	crab
lion	goat	wren	frog	frog	frog	crow	lion	wolf	crow
goat	duck	lynx	gnat	goat	goat	deer	goat	mule	deer
mole	frog	crab	goat	lion	hawk	dove	duck	mole	dove
frog	dove	swan	hawk	mole	lion	duck	frog	puma	duck
swan	clam	bear	lion	clam	lynx	frog	dove	seal	frog
clam	hawk	clam	lynx	hawk	mole	gnat	clam	deer	gnat
hawk	deer	bass	lynx	swan	swan	goat	hawk	lion	goat
wren	crow	goat	mole	wren	wren	hawk	deer	goat	hawk
mule	gnat	mule	mule	gnat	gnat	mule	crow	bear	lion
oryx	lynx	oryx	oryx	lynx	lynx	oryx	lynx	lynx	lynx
gnat	lynx	gnat	swan	mule	mule	lynx	lynx	gnat	lynx
lynx	puma	lynx	wren	oryx	oryx	lynx	oryx	lynx	mole
puma	worm	puma	puma	crow	puma	puma	puma	frog	mule
worm	seal	worm	worm	puma	worm	worm	worm	crab	oryx
seal	oryx	seal	seal	seal	crow	seal	seal	bass	puma
crow	mule	crow	crow	worm	deer	lion	mule	crow	seal
deer	wolf	deer	deer	deer	dove	wren	wren	clam	swan
wolf	wren	wolf	wolf	dove	duck	wolf	wolf	hawk	wolf
dove	swan	dove	dove	duck	seal	mole	swan	dove	worm
duck	mole	duck	duck	wolf	wolf	swan	mole	duck	wren
----	----	----	----	----	----	----	----	----	----
0									1

(0) Original input

(1) Sorted

(2) Selection sort

(3) Insertion sort

(4) Mergesort (*top-down*)

(5) Mergesort (*bottom-up*)

(6) Quicksort (*standard, no shuffle*)

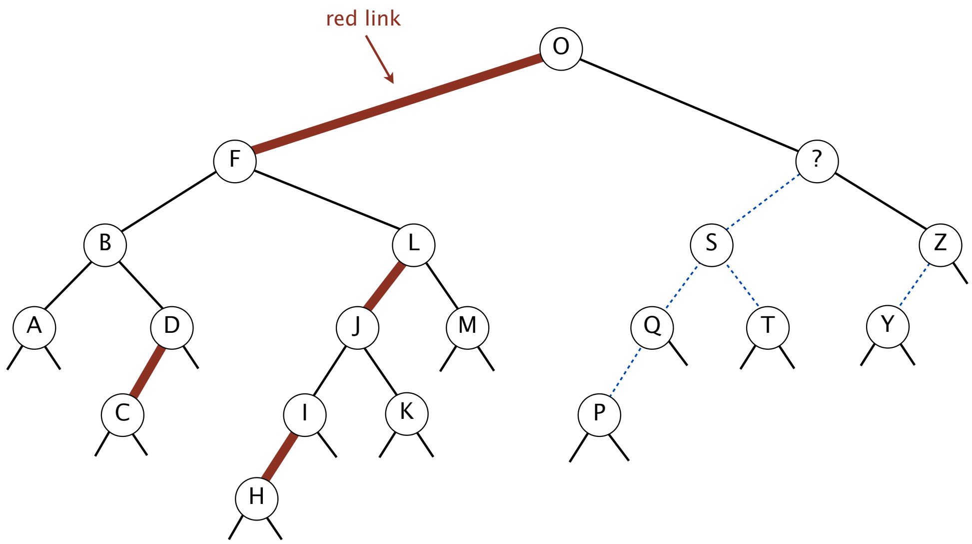
(7) Quicksort (*3-way, no shuffle*)

(8) Heapsort

(9) Knuth shuffle

5. **Red-black BSTs. (8 points)**

Consider the following left-leaning red-black BST. Some of the colors and key values are suppressed.



(a) Which one or more of the keys below could be the one labeled with a question mark?

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

(b) Which one or more of the keys below *must* be red (link between it and its parent is red)?

P Q S T Y

(c) How many *left rotation*, *right rotation*, and *color flip* operations would be used to insert each key below into the *original* red-black BST above?

	E	N	G
rotateLeft()			
rotateRight()			
flipColors()			

6. **Hashing. (6 points)**

Suppose that the following keys are inserted in some order into an initially empty linear-probing hash table of size 7 (assuming no resizing), using the following table of hash values:

key	hash
A	5
B	2
C	5
D	1
E	4
F	1
G	3

(a) Give the contents of the linear-probing array if the keys are inserted in alphabetical order: A, B, C, D, E, F, G.

0	1	2	3	4	5	6

(b) Which of the following could be the contents of the linear-probing array if the keys are inserted in some other order?

I.

0	1	2	3	4	5	6
A	F	D	B	G	E	C

II.

0	1	2	3	4	5	6
F	A	D	B	G	E	C

III.

0	1	2	3	4	5	6
C	A	B	G	F	E	D

Circle the best answer.

(a) I only.

(b) I and II only.

(c) I and III only.

(d) I, II and III.

(e) None.

7. **Comparing two arrays of points. (8 points)**

Given two arrays `a[]` and `b[]`, containing M and N distinct points in the plane, respectively, (with $N \geq M$), design an algorithm to determine how many points are in common between the two arrays. The running time of your algorithm should be proportional to $N \log M$ in the worst case and use at most a constant amount of extra memory.

Partial credit for $N \log N$ or for using a linear amount of extra memory.

(a) Give a crisp and concise English description of your algorithm in the box below. Your answer will be graded on correctness, efficiency, clarity, and conciseness.

(b) What is the order of growth of the *worst case* running time of your algorithm? Circle the best answer.

M N $M \log M$ $M \log N$ $N \log M$ $N \log N$ MN N^2

(c) How much *extra memory* does your algorithm use? Circle the best answer.

1 $\log M$ $\log N$ M N MN N^2

8. **Randomized priority queue. (8 points)**

Describe how to add the methods `sample()` and `delRandom()` to our binary heap implementation of the `MinPQ` API. The two methods return a key that is chosen uniformly at random among the remaining keys, with the latter method also removing that key.

`public class MinPQ<Key extends Comparable<Key>>`

	`MinPQ()`	*create an empty priority queue*
`void`	`insert(Key key)`	*insert a key into the priority queue*
`Key`	`min()`	*return the smallest key*
`Key`	`delMin()`	*return and remove the smallest key*
`Key`	`sample()`	*return a key that is chosen uniformly at random*
`Key`	`delRandom()`	*return and remove a key that is chosen uniformly at random*

You should implement the `sample()` method in constant time and the `delRandom()` method in time proportional to $\log N$, where N is the number of keys in the data structure. For simplicity, do not worry about resizing the underlying array.

Your answer will be graded on correctness, efficiency, clarity, and conciseness.

- `sample()`:

- `delRandom()`: