COS 226 Algorithms and Data Structures Spring 2004

Midterm

This test has 6 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	
Sub 1	

Problem	Score
4	
5	
6	
Sub 2	

Total	

Name:

Login ID:

Precept: 1 12:30
2 1:30
4 3:30

1. **9 sorting algorithms. (10 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 9 sorting algorithms listed below. Match up each algorithm by writing its number under the corresponding column. Use each number exactly once.

bold	abet	abet	coma	acme	ache	slaw	abet	bold	abet
cash	acid	ache	gala	akin	acne	slab	acid	cash	ache
coma	acme	acid	idea	abet	ajar	fame	acme	acre	acid
band	akin	acme	club	acid	band	lady	acne	band	acme
club	band	acne	slab	ajar	bold	fall	ajar	ache	acne
acme	bold	acre	bold	acne	acme	dawn	akin	acme	acre
akin	cash	ajar	band	also	akin	fake	band	akin	ajar
abet	club	akin	acid	ache	abet	idea	bold	abet	akin
knee	coma	also	bald	acre	bank	knee	cart	bank	also
slab	fall	bald	acme	bold	bald	club	cash	bald	bald
fall	knee	band	knee	band	acre	debt	cell	chef	band
acid	slab	bank	fame	bald	acid	city	club	acid	bank
fame	fame	bold	acne	bank	also	bold	coma	also	fame
ajar	ajar	cash	ache	cash	cell	else	fall	ajar	bold
cart	cart	coma	else	coma	chef	coma	fame	cart	cart
cell	cell	cart	fake	club	city	cell	gala	cell	cell
acne	acne	club	acre	cart	chug	dark	knee	acne	club
gala	gala	cell	half	cell	cash	gala	slab	chug	gala
half	half	chef	chef	chef	coma	half	ache	half	half
also	also	chug	chug	city	club	also	acre	fame	knee
chef	chef	knee	cash	chug	cart	chef	also	fall	chef
debt	debt	slab	bank	debt	debt	cash	bald	debt	debt
bald	bald	fall	dark	dawn	slab	bald	bank	slab	slab
bank	bank	city	fall	dark	knee	bank	chef	knee	coma
city	city	fame	cell	else	idea	acid	chug	city	city
ache	ache	dark	akin	fall	lady	ache	city	club	cash
dawn	dawn	dawn	dawn	fame	dawn	acme	dark	dawn	dawn
else	else	debt	also	fake	else	akin	dawn	else	else
slaw	slaw	else	ajar	gala	slaw	ajar	debt	slaw	slaw
fake	fake	gala	abet	half	fake	cart	else	fake	fake
acre	acre	half	cart	idea	fall	acre	fake	coma	fall
idea	idea	fake	debt	knee	fame	abet	half	idea	idea
lady	lady	idea	slaw	lady	half	band	idea	lady	lady
dark	dark	lady	city	slab	gala	acne	lady	dark	dark
chug	chug	slaw	lady	slaw	dark	chug	slaw	gala	chug
----	----	----	----	----	----	----	----	----	----
0									

(0) Original input

(1) 3-way radix quicksort

(2) Bubble sort

(3) Heap sort

(4) Insertion sort

(5) LSD radix sort

(6) Mergesort

(7) MSD radix sort

(8) Quicksort

(9) Selection sort

2. **Analysis of algorithms. (6 points)**

Each of the Java functions on the left take a nonnegative integer n as input, and returns a string of length $N = 2^n$ with all a's. Choose the best matching asymptotic complexity bound on the right. Recall that concatenating two strings in Java takes time proportional to the sum of their lengths.

A. $\log N$

B. N

C. $N \log N$

D. N^2

E. 2^N

___
```
public static String method1(int n) {
   String s = "a";
   for (int i = 0; i < n; i++)
      s = s + s;
   return s;
}
```

___
```
public static String method2(int n) {
   String s = "";
   int N = 1 << n;  // 2^n
   for (int i = 0; i < N; i++)
       s = s + "a";
   return s;
}
```

___
```
public static String method3(int n) {
   if (n == 0) return "a";
   else return method3(n-1) + method3(n-1);
}
```

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

Insert the following keys into a linear probing hash table of size $M = 7$, using the hash function $f(x) = i \ \% \ 7$, where x is the ith letter of the alphabet.

x	P	R	O	B	I	N	G
i	16	18	15	2	9	14	7
f(i)	2	4	1	2	2	0	0

4. **Markov model. (10 points)**

You don't actually need to understand this paragraph to answer the question, it's purely motivation. Professor Carlo wants to simulate a gas reacting with the surface of a substrate on which chemical reactions occur at different rates. The *kinetic Monte Carlo method* discretizes the substrate into N sites, and assigns probabilities p_i (according to the laws of statistical mechanics) that the next reaction will occur at site i.

(a) Given an array of N positive real number p_i that sum to 1, give an $O(N)$ algorithm for selecting a site i at random, so that site i is chosen with probability p_i. To illustrate your algorithm, you may use the following 8 values.

p0	p1	p2	p3	p4	p5	p6	p7
0.14	0.04	0.30	0.11	0.13	0.15	0.10	0.03

(b) Professor Carlo wants N to be large (say, billions) and must choose sites at random many times (say, tens of billions). Explain how to do this using $O(N \log N)$ time for a one-time preprocessing step, and $O(\log N)$ time per random sample.

(c) Suppose that after each step, Professor Carlo must update a small number (say, at most 4) of the probabilities p_i. Explain how to create a data structure that takes $O(N \log N)$ time for preprocessing and $O(\log N)$ time per random sample.

5. **Desecrated quicksort. (12 points)**

A distinguished professor at a prominent northeastern university teaches the following implementation of quicksort. (We have translated from VB.NET to Java, and included comments to show the author's intent.)

```
public static void desecratedQuicksort(double[] a) {
   int N = a.length;                                    // # elements
   int hi = 0;                                          // # high elements
   int lo = 0;                                          // # low  elements
   double[] H = new double[N];                          // high elements
   double[] L = new double[N];                          // low  elements

   for (int i = 1; i < N; i++) {                        // partition on a[0]
       if (a[i] > a[0]) H[hi++] = a[i];                 // high piece
       else             L[lo++] = a[i];                 // low  piece
   }

   if (lo > 0) {                                        // if non-empty
       double[] temp = new double[lo-1];                // resize
       for (int i = 0; i < lo-1; i++) temp[i] = L[i];   // copy low piece
       desecratedQuicksort(temp);                       // sort low piece
       a[lo] = a[0];                                    // partition element
       for (int i = 0; i < lo-1; i++) a[i] = temp[i];   // copy back
   }

   if (hi > 0) {
       double[] temp = new double[hi-1];
       for (int i = 0; i < hi-1; i++) temp[i] = H[i];
       desecratedQuicksort(temp);
       for (int i = 0; i < hi-1; i++) a[lo+1+i] = temp[i];
   }
}
```

Critique the professor's version of quicksort. Pay specific attention to correctness, average and worst case running time, average and worst case memory usage, and stability. Justify your answers with concise and precise explanations.

(a) Correctness

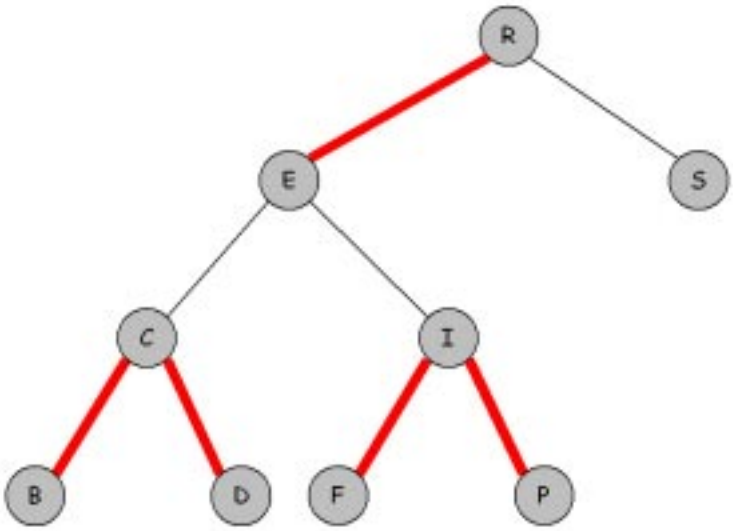
(b) Running time.

(c) Memory usage.

(d) Stability.

(e) Write and sign the following solemn oath: *I hereby pledge that I will never desecrate quicksort in such a grotesque manner.*

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



Insert the key T, then the key Q into the red-black tree above. Draw the resulting red-black tree in the space below.