COS 226 **Algorithms and Data Structures** **Fall 2012**

Midterm

This test has 9 questions worth a total of 55 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:

Room:

Precept:

P01	F 11	Maia Ginsburg
P02	F 12:30	Diego Perez Botero
P03	F 1:30	Diego Perez Botero
P03B	F 1:30	Dushyant Arora
P04	Th 2:30	Maia Ginsburg
P04A	Th 2:30	Dan Larkin

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

   In the space provided on the front of the exam, write your name and Princeton NetID; circle your precept number; write the name of the room in which you are taking the exam; and write and sign the honor code.

1. **Union find. (4 points)**

   Circle the letters corresponding to `id[]` arrays that *cannot* possibly occur during the execution of the weighted quick union algorithm.

		0	1	2	3	4	5	6	7	8	9
A.	a[i]:	8	0	4	0	0	4	0	4	2	0
B.	a[i]:	4	1	8	2	1	5	1	1	4	5
C.	a[i]:	3	3	6	9	3	6	3	4	1	9
D.	a[i]:	2	1	1	1	1	1	1	2	1	7

2. **Eight sorting 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 strings in sorted order; the other columns are the contents at some intermediate step during one of the algorithms listed below. Match up each algorithm by writing its number under the corresponding column. Use each number exactly once.

ENYA	ABBA	INXS	DIDO	BUSH	BECK	ABBA	BLUR	ABBA	ABBA
KISS	ACDC	HOLE	CARS	DIDO	DIDO	ACDC	ABBA	ACDC	ACDC
INXS	BECK	FUEL	BECK	CARS	CARS	BUSH	DIDO	BLUR	BECK
STYX	BLUR	ENYA	ACDC	ENYA	ACDC	ENYA	FUEL	BUSH	BLUR
SOAD	BUSH	ENYA	BLUR	BECK	BUSH	FUEL	BUSH	ENYA	BUSH
ACDC	CARS	DIDO	ABBA	ACDC	ABBA	INXS	ACDC	FUEL	CARS
KORN	DIDO	BUSH	BUSH	BLUR	BLUR	KISS	ENYA	INXS	DIDO
FUEL	ENYA	BECK	ENYA	ABBA	ENYA	KORN	HOLE	KISS	ENYA
BUSH	ENYA	ABBA	ENYA	ENYA	INXS	PINK	ENYA	KORN	ENYA
ABBA	FUEL	ACDC	WHAM	FUEL	KISS	SOAD	BECK	MUSE	FUEL
WHAM	HOLE	CARS	PINK	WHAM	ENYA	STYX	INXS	PINK	HOLE
PINK	INXS	BLUR	FUEL	PINK	PINK	WHAM	KORN	SOAD	INXS
BLUR	STYX	JAYZ	MUSE	KORN	KORN	BECK	SADE	STYX	JAYZ
MUSE	MUSE	KISS	KORN	MUSE	MUSE	BLUR	CARS	WHAM	KISS
BECK	PINK	KORN	MOBY	SOAD	FUEL	HOLE	JAYZ	BECK	KORN
MOBY	MOBY	MOBY	HOLE	MOBY	MOBY	MOBY	MOBY	MOBY	MOBY
HOLE	WHAM	MUSE	TSOL	HOLE	HOLE	MUSE	SOAD	HOLE	MUSE
TSOL	TSOL	PINK	JAYZ	TSOL	JAYZ	TSOL	MUSE	TSOL	PINK
JAYZ	JAYZ	RUSH	SOAD	JAYZ	RUSH	CARS	KISS	JAYZ	RUSH
ENYA	SOAD	SADE	SADE	STYX	SADE	DIDO	PINK	ENYA	SADE
SADE	SADE	SOAD	STYX	SADE	WHAM	ENYA	STYX	SADE	SOAD
CARS	KISS	STYX	INXS	INXS	SOAD	JAYZ	TSOL	CARS	STYX
DIDO	KORN	TSOL	RUSH	KISS	STYX	RUSH	RUSH	DIDO	TSOL
RUSH	RUSH	WHAM	KISS	RUSH	TSOL	SADE	WHAM	RUSH	WHAM
----	----	----	----	----	----	----	----	----	----
0									1

(0) Original input

(1) Sorted

(2) Selection sort

(3) Insertion sort

(4) Shellsort
(*13-4-1 increments*)

(5) Mergesort
(*top-down*)

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

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

(8) Quicksort
(*dual-pivot, no shuffle*)

(9) Heapsort

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

Suppose that you have an array of length $2N$ consisting of N B's followed by N A's. Below is the array when $N = 10$.

```
B B B B B B B B B B A A A A A A A A A A
```

(a) How many compares does it take to insertion sort the array as a function of N? Use tilde notation to simplify your answer.

(b) How many compares does it take to 3-way quicksort the array as a function of N (using Dijsktra's 3-way partitioning algorithm)? Use tilde notation to simplify your answer.

4. **3-heaps. (8 points)**

A *3-heap* is an array representation of a complete ternary tree, where the key in each node is greater than (or equal to) the keys in each of its children.

(a) Perform a *delete-the-maximum* operation on the following 3-heap, which is the level-order traversal of a complete ternary tree, using 1-based indexing.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
--	88	33	77	66	10	30	25	23	60	75	14	21	50	9	7

Fill in the table below to show the resulting 3-heap, circling any entries that change.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
															--

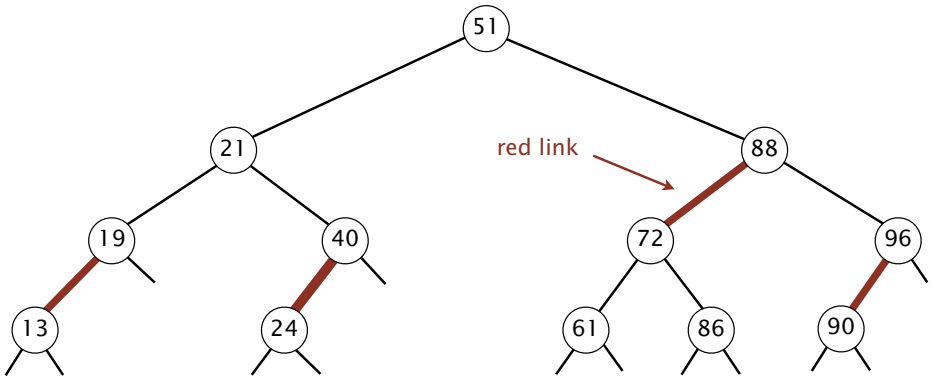
(b) Given the array index k of a key, what are the indices of its three (potential) children as a function of k? Assume 1-based indexing and circle your three answers.

(c) What is the maximum number of compares for a *delete-the-maximum* operation as a function of the number of keys N in the data structure? Circle the best answer.

~ 1 $\sim \log_2 N$ $\sim \log_3 N$ $\sim 2 \log_3 N$ $\sim 2 \log_2 N$ $\sim 3 \log_3 N$ $\sim N$

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

Consider the following left-leaning red-black BST.



Insert the key `99` into the red-black BST and give the level-order traversal of the resulting BST. Circle the keys who parent link is red.

6. **Problem identification. (7 points)**

   You are applying for a job at a new software technology company. Your interviewer asks you to identify the following tasks as either possible (with algorithms and data structures learned in this course), impossible, or an open research problem. You may use each letter once, more than once, or not at all.

I. Impossible

P. Possible

O. Open

_____ Given any array of N distinct integers, determine whether there are three integers that sum to exactly zero in time proportional to $N^{1.5}$.

_____ Given any array of N distinct integers, determine whether there are three integers that sum to exactly zero in time proportional to N^2.

_____ Implement a FIFO queue with a constant amount of memory plus two LIFO stacks, so that each queue operation uses a constant amortized number of stack operations.

_____ Given any left-leaning red-black BST containing N keys, find the largest key less than or equal to a given key in logarithmic time.

_____ Design a priority queue implementation that performs *insert*, *max*, and *delete-max* in $\sim \frac{1}{3} \lg N$ compares per operation, where N is the number of comparable keys in the data structure.

_____ Given any array of N keys containing three distinct values, sort it in time proportional to N and using only a constant amount of extra space.

_____ Design a practical, in-place, stable, sorting algorithm that guarantees to sort any array of N comparable keys in at most $\sim N \lg N$ compares.

7. **LRU cache. (8 points)**

An *LRU cache* is a data structures that stores up to N *distinct* keys. If the data structure is full when a key not already in the cache is added, the LRU cache first removes the key that was *least recently cached*.

Design a data structure that supports the following API:

`public class LRU<Key>`	
`LRU(int N)`	*create an empty LRU cache with capacity N*
`void cache(Key key)`	*if there are N keys in the cache and the given key is not already in the cache, (i) remove the key that was least recently used as an argument to* `cache()` *and (ii) add the given key to the LRU cache*
`boolean inCache(Key key)`	*is the key in the LRU cache?*

The operations `cache()` and `inCache()` should take constant time on average under the uniform hashing assumption.

For example,

```
LRU<String> lru = new LRU<String>(5);
                                 // LRU cache   (in order of when last cached)
lru.cache("A");                  // A           (add A to front)
lru.cache("B");                  // B A         (add B to front)
lru.cache("C");                  // C B A       (add C to front)
lru.cache("D");                  // D C B A     (add D to front)
lru.cache("E");                  // E D C B A   (add E to front)
lru.cache("F");                  // F E D C B   (remove A from back; add F to front)
boolean b1 = lru.inCache("C");   // F E D C B   (true)
boolean b2 = lru.inCache("A");   // F E D C B   (false)
lru.cache("D");                  // D F E C B   (move D to front)
lru.cache("C");                  // C D F E B   (move C to front)
lru.cache("G");                  // G C D F E   (remove B from back; add G to front)
lru.cache("H");                  // H G C D F   (remove E from back; add H to front)
boolean b3 = lru.inCache("D");   // H G C D F   (true)
```

Give a crisp and concise English description of your data structure and how the `inCache()` *and* `cache()` *operation are implemented. Your answer will be graded on correctness, efficiency, and clarity.*

- Describe your data structure(s). For example, if you use a linear probing hash table, specify what are the hash table key-value pairs. Also, show (with a small diagram) your data structure(s) immediately after the sequence of operations on the previous page.

- `inCache(Key key):`

- `cache(Key key):`

8. **Detecting a duplicate. (8 points)**

Given k sorted arrays containing N keys in total, design an algorithm that determine whether there is any key that appears more than once.

Your algorithm should use extra space at most proportional to k. For full credit, it should run in time at most proportional to $N \log k$ in the worst case; for partial credit, time proportional to Nk.

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

(b) What is the order of growth of the worst case running time of your algorithm as a function of N and k? Briefly justify your answer.

N $k \log N$ $N \log k$ $N \log N$ Nk $Nk \log k$ $Nk \log N$ N^2