Princeton University – Computer Science
COS226: Data Structures and Algorithms

Midterm, Spring 2013

This test has 9 questions worth a total of 71 points. The exam is closed book, except that you are allowed to use a one page written cheat sheet. No calculators or other electronic devices are permitted. Give your answers and show your work in the space provided. Write 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."

	Score		Score
1		6	
2		7	
3		8	
4		9	
5			
Sub 1		Sub 2	

Total	/71

Name:

Login ID:

Exam Room: McCosh 10
McCosh 62
McCosh 66
East Pyne 10

P01	Josh	Th 11	P05	Jennifer	F 11	P07	Nico	F 230
P02	Maia	Th 1230	P05A	Stefan	F 11	P08	Maia	F 10
P03	Arvind	Th 130	P06	Diego	F 230			
P04	Diego	F 330	P06A	Dushyant	F 230			

Tips:

- There may be partial credit for incomplete answers. Write as much of the solution as you can, but bear in mind that we will deduct points if your answers are more complicated than necessary.
- There are a lot of problems on this exam. Work through the ones with which you are comfortable first. Do not get overly captivated by interesting design issues.
- Not all information provided in a problem may be useful.

Optional. Mark along the line to show your feelings on spectrum between ☹ and ☺.

Before exam: [☹_____☺].
After exam: [☹_____☺].

1. **Analysis of Algorithms (6 points).** The code below operates on bacterial genomes of approximately 1 megabyte in size.

```
int N = Integer.parseInt(args[0]);
String[] genomes = new String[N];
for (int i = 0; i < N; i++) {
     In gfile = new In("genomeFile" + i + ".txt");
     genomes[i] = gfile.readString();
}

for (int i = 1; i < N; i++) {
     for (int j = i; j > 0; j--) {
          if (genomes[j-1].length() > genomes[j].length())
               exch(genomes, j-1, j);
          else break;
     }
}
```

(a) What is the theoretical ***order of growth*** of the worst case running time as a function of N?

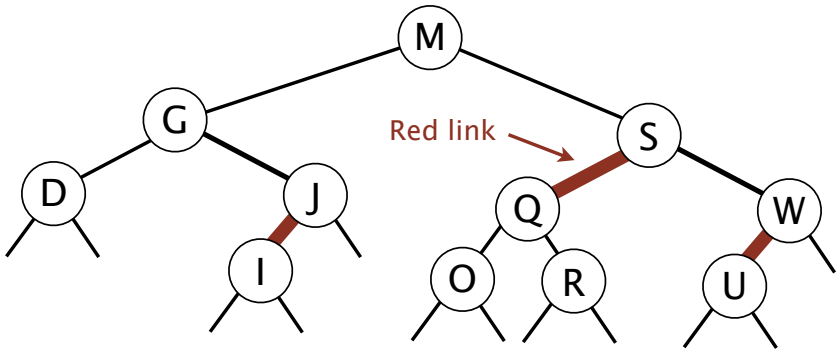
(b) A table of runtimes for the program above is given below. ***Approximate*** the empirical run time in ***tilde notation*** as a function of N. Do not leave your answer in terms of logarithms.

N	Time (s)
1	0.15
2	0.14
4	0.19
8	0.41
16	0.85
32	1.66
64	3.38

Answer:

(c) Explain any discrepancy between your answers to (a) and (b). Be as specific and detailed as possible.

2. Red-black BSTs (8 points). Consider the Red-Black tree below.



(a) Circle the keys whose insertion will cause either rotations and/or color flips.

A B C E F H K L N P T V X Y Z

(b) Draw ***and give the level order traversal*** of the tree that results after inserting the key Z.

Write the level order traversal in the box below. **Clearly indicate red nodes by drawing an upward pointing arrow below any red nodes.**

(c) In plain English, describe the function of `mystery()`.

```
public boolean mystery() {
    return mystery(root, null, null);
}

private boolean mystery(Node x, Key a, Key b) {
    if (x == null) return true;
    if (a != null && x.key.compareTo(a) <= 0) return false;
    if (b != null && x.key.compareTo(b) >= 0) return false;
    return mystery(x.left, a, x.key) &&
           mystery(x.right, x.key, b);
}
```

Draw anything you'd like in the space below:

3. Symbol Tables (8 points).

(a) For the symbol table applications below, pick the best symbol table implementation from the list on the right.

----- Lookup table for computing sin(theta), where theta is one of 1,000,000 possible angles spaced evenly between 0 and π.

----- Database that maps sound data (from a file) to artist name.

----- Fastest guaranteed insert, delete and search for an arbitrary numerical data set.

A. Standard BST

B. Red black BST

C. Hash table

D. Ordered Array

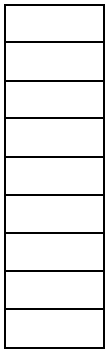
E. Unordered Array

F. Heap

(b) Consider a hash table of **fixed size** 9 with hash function h(k)=k mod 9, where the following (key, value) pairs are inserted in the given order:

(37,'A'), (13,'B'), (71,'C'), (25,'D'), (53,'E'), (7,'F')

Draw the table that results if collisions are handled through separate chaining:



Key	Value	Hash
37	A	1
13	B	4
71	C	8
25	D	7
53	E	8
7	F	7

(c) Show the array that results if collisions are handled using linear probing.

(d) Consider an initially empty **Symbol Table** implemented using a hash table of size M with hash function h(k)=k mod M. **In the worst case** for any possible sequence of inputs where N > M, what is the order of growth of inserting N (key, value) pairs with distinct keys into the table if **separate chaining** is used to resolve collisions?

Suppose that each bucket of the table stores an unordered linked list. When adding a new element to an unordered linked list, each element is inserted at the beginning of the list.

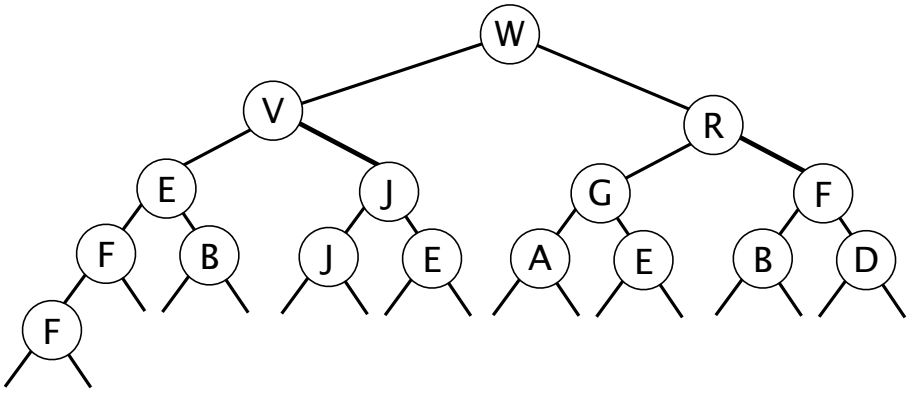
4. Quicksort (5 points).

(a) Show the results after 3-way partitioning on S.

S W I M P E A T S F R I E S

(b) Give a very short proof in plain English that 3-way quicksort always completes in linear time for an array with N items and 7 distinct keys.

5. Heaps and Priority Queues (7 points).



(a) Give the array that represents the correct max heap after deleting the max [ignore the bug in the heap where the 'E' should be something larger than F but less than V].

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
-																	

(b) Consider the process of creating a max heap from an arbitrary text file containing N integers. What is the order of growth of the run time of the best possible algorithm for this process?

(c) Is there any reason in terms of time or space efficiency to use a max heap to implement a max priority queue instead of using a left leaning red black tree? Justify your answer.

6. Sorting (8 points). The leftmost column is the original input of strings to be sorted; the rightmost column gives the strings in sorted order; the other columns are the contents at the intermediate step during one of the 8 sorting algorithms listed below. Match up each algorithm by writings its number under the corresponding column. Use each number exactly once.

HELP	AMTR	EASE	HELP	DINS	AMTR	WATC	AMTR	ETYP	AMTR
IFYO	APPE	ETYP	IFYO	ETYP	APPE	SWAR	APPE	AREW	APPE
UARE	DINS	AREW	READ	AMTR	DINS	UARE	AREW	EVIL	AREW
READ	EASE	HELP	UARE	APPE	HELP	SEND	DINS	AMTR	DINS
INGT	HELP	EVIL	AMTR	HELP	HISI	SORT	EASE	APPE	EASE
HISI	HISI	HELP	APPE	HELP	IDEA	THEP	ETYP	DINS	ETYP
AMTR	IDEA	AMTR	HISI	AREW	IFYO	RATS	EVIL	EASE	EVIL
APPE	IFYO	APPE	INGT	EASE	INGA	READ	HELP	HELP	HELP
DINS	INGA	DINS	DINS	HELP	INGT	RATS	HELP	HELP	HELP
IDEA	INGT	HELP	IDEA	IDEA	READ	RATS	HELP	HELP	HELP
SORT	ITHM	SORT	INGA	HISI	SORT	MALL	HISI	INGA	HISI
INGA	LGOR	INGA	SORT	EVIL	UARE	OVER	IDEA	LGOR	IDEA
LGOR	OHPL	LGOR	EASE	LACE	LGOR	LGOR	LGOR	ITHM	IFYO
ITHM	READ	ITHM	ITHM	IFYO	ITHM	ITHM	ITHM	OHPL	INGA
OHPL	SORT	OHPL	LGOR	INGT	OHPL	OHPL	OHPL	SORT	INGM
EASE	UARE	IDEA	OHPL	INGA	EASE	ETYP	INGT	SEND	INGT
SEND	EVIL	SEND	EVIL	RATS	SEND	APPE	SEND	IDEA	ITHM
HELP	HELP	HISI	HELP	INGM	HELP	HELP	IFYO	RATS	LACE
RATS	MALL	RATS	RATS	ITHM	RATS	DINS	RATS	HISI	LGOR
EVIL	OVER	INGT	SEND	RATS	EVIL	EVIL	READ	RATS	MALL
RATS	RATS	RATS	MALL	SEND	RATS	IDEA	RATS	SWAR	OHPL
SWAR	RATS	SWAR	OVER	LGOR	SWAR	INGT	SWAR	MALL	OVER
MALL	SEND	MALL	RATS	MALL	MALL	IFYO	MALL	OVER	RATS
OVER	SWAR	OVER	SWAR	RATS	OVER	HISI	OVER	THEP	RATS
THEP	AREW	THEP	HELP	THEP	THEP	INGA	THEP	LACE	RATS
LACE	ETYP	LACE	LACE	SWAR	LACE	LACE	LACE	INGT	READ
HELP	HELP	READ	RATS	OHPL	HELP	HELP	INGA	RATS	SEND
RATS	INGM	RATS	THEP	READ	RATS	HELP	RATS	READ	SORT
AREW	LACE	UARE	AREW	UARE	AREW	AREW	UARE	WATC	SWAR
WATC	RATS	WATC	ETYP	WATC	WATC	AMTR	WATC	INGM	THEP
INGM	THEP	INGM	INGM	OVER	INGM	INGM	INGM	UARE	UARE
ETYP	WATC	IFYO	WATC	SORT	ETYP	EASE	SORT	IFYO	WATC
----	----	----	----	----	----	----	----	----	----
0									1

0: Original input
1: Sorted
2: Selection sort
3: Insertion sort
4: Shellsort (13-4-1)
5: Mergesort (top-down)
6: Mergesort (bottom-up)
7: Quicksort (standard, no shuffle)
8. Quicksort (3-way, no shuffle)
9. Heapsort

7. You can't do that on television (10 points). Identify the following as possible, impossible, or open.

P. Possible

I. Impossible

O. Open

----- Sorting any arbitrary N comparable items with a constant amount of extra space in worst case time proportional to N.

----- Median identification for an arbitrary array in average case time proportional to N.

----- Median identification for an arbitrary array in worst case time proportional to N.

----- Building a left leaning red black tree with height 3 lg N.

----- Building a binary search tree with height `floor(lg N)` from an unsorted array in linearithmic worst case time.

----- Building a symbol table that uses 16-bit integers as keys and is guaranteed to complete insertion, deletion, and search operations in constant time for any possible key/value pairs.

----- Finding a worst-case sequence of N insertions that causes a left leaning red black (LLRB) tree to take N^2 time to construct.

----- Heapification of any array in linear time using only sink operations.

----- Heapification of any array in linear time using only swim operations.

----- Occurrence of the array [1 1 1 1 0 9 1 7 5 0] during execution of the weighted quick union algorithm.

8. Addendum (9 points). The `addBlock()` operation is used to add M `comparables` to an existing sorted data set of N `comparables`, where M << N. A data set of size N is considered sorted if it can be iterated through in sorted order in N time.

COS226 student Frankie Halfbean makes two choices. First, he selects a sorted array as the data structure. Secondly, he selects insertion sort as the core algorithm, explaining that insertion sort is very fast for almost sorted arrays. To add a new block of M comparables, the algorithm simply creates an array of length N+M, copies over the old N values into the new array, copies over the new M values to the end of the array, and finally insertion sort is used to bring everything into order. The old array is left available for garbage collection.

(a) What is the worst case order of growth of the run time as a function of N and M?

(b) Design a scheme that has a better order of growth for the run time in the worst case. For full credit, design a scheme that uses optimal space and time to within a constant factor.

9. ExcavatingDeque (10 points).

An ExcavatingDeque behaves exactly like the Deque from assignment 2, except for the `removeFirst()` and `removeLast()` operations. If the item to be removed appears in the Deque more than once, then the duplicate with the *greatest nesting depth* is removed from the Deque instead of the one at the end. The *nesting depth* of an item X is the ***minimum number of items equal to X*** that must be crossed to reach either edge of the Deque. For example, if the ExcavatingDeque is given by 7 ↔ **7** ↔ 8 ↔ 7 ↔ 1 ↔ 3 ↔ 2, then the bolded 7 has the greatest nesting depth, because the path to both edges involves crossing another 7. If removeFirst() were called, then the bolded 7 would be removed instead of the one at the front. If there is a tie, then either may be removed.

```
public class ExcavatingDeque {
      ExcavatingDeque()
      void addFirst(int x)
      void addLast(int x)
      int removeFirst()
      int removeLast()
}
```

All operations should complete **in constant time**. For partial credit, complete all operations in N log d time, where d is the number of times the item appears in the ExcavatingDeque. Your ExcavatingDeque should **use memory proportional to the number of items**. You may assume the uniform hashing assumption.

Longer example:

`addFirst(4)`	4
`addFirst(3)`	3 ↔ 4
`addFirst(6)`	6 ↔ 3 ↔ 4
`addFirst(7)`	7 ↔ 6 ↔ 3 ↔ 4
`addFirst(6)`	6 ↔ 7 ↔ 6 ↔ 3 ↔ 4
`addFirst(7)`	7 ↔ 6 ↔ 7 ↔ 6 ↔ 3 ↔ 4
`addFirst(8)`	8 ↔ 7 ↔ 6 ↔ 7 ↔ 6 ↔ 3 ↔ 4
`addFirst(6)`	6 ↔ 8 ↔ 7 ↔ 6 ↔ 7 ↔ 6 ↔ 3 ↔ 4
`removeFirst()`	6 ↔ 8 ↔ 7 ↔ 7 ↔ 6 ↔ 3 ↔ 4
`removeLast()`	6 ↔ 8 ↔ 7 ↔ 7 ↔ 6 ↔ 3
`removeFirst()`	//same nesting depth so either 6 may be removed [your choice]

On the next page, give a crisp and concise English description of your data structure and how the addFirst() and removeFirst() operation are implemented. Your answer will be graded on correctness, efficiency, and clarity.

- Describe your the data structure(s) you'd use to implement the ExcavatingDeque class. For example, if you use a linear probing hash table, specify the hash table key-value pairs. Show (with a small diagram) your data structure(s) after addFirst is called with 5, 7, 7, 8, 7, 1, 3, 2 as arguments.

- `addFirst()`:

- `removeFirst()`:

Bonus question (no credit). What famous computer scientist climbed in an oven for fun?