CSE 332: Data Abstractions — Autumn 2015

Practice Midterm Exam

Name:	
ID #:	
TA:	
Section:	

INSTRUCTIONS:

- You have **50 minutes** to complete the exam.
- The exam is closed book and closed notes. You may not use cell phones or calculators.
- All answers you want graded should be written on the exam paper.
- If you need extra space, use the back of a page.
- The problems are of varying difficulty.
- If you get stuck on a problem, move on and come back to it later.
- It is to your advantage to read all the problems before beginning the exam.

Problem	Points	Score	Problem	Points	Score
1	8		6	15	
2	3		7	8	
3	3		8	15	
4	6		9	10	
5	6		10	15	
			Σ	89	

One Liners.

This section has questions that require very short answers. To get full credit, you should answer in no more than one sentence per question.

1. X Marks The Spot [8 points]

For each of the following rows, circle each option on the right that is true for the function on the left and X each option that is false for the function on the left.

$\sum_{i=0}^{n} i^2$	$\mathcal{O}(n)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^3)$	$\mathcal{O}(n^4)$
3^n	$\mathcal{O}(3^n)$	$\mathcal{O}(3^{n/2})$	$\Omega(3^n)$	$\Omega(3^{n/2})$
$10000n^{25}$	$\mathcal{O}(n)$	$\mathcal{O}(n^{26})$	$\Omega(n)$	$\Omega(n^{26})$
$\log n$	$\mathcal{O}(\log n)$	$\mathcal{O}(\log\log n)$	$\Omega(\log n)$	$\Omega(\log\log n)$
n^n	$\mathcal{O}(20000^n)$	$\Omega(20000^n)$	$\Theta(20000^n)$	$\Theta(n^n + 20000^n)$

2. `find` The Error [3 points]

Consider a *dictionary* implemented using a sorted array. Consider the following argument:

> We claim that `find()` is amortized $\mathcal{O}(1)$ in this implementation. Consider a sequence of $\lg(n)$ operations. Each operation takes $\lg(n)$ time; so, the amortized cost of each operation is $\frac{\lg(n)}{\lg(n)} = \mathcal{O}(1)$.

This argument is faulty. Explain why it's faulty and give a correct (tight) asymptotic bound for the actual amortized cost.

3. It's $\mathcal{O}$lementary, My Dear Watson [3 points]

Let $f : \mathbb{N} \to \mathbb{N}$ and $g : \mathbb{N} \to \mathbb{N}$ be increasing functions with $f(n) \neq g(n)$ for any $n \in \mathbb{N}$. Consider the statement:

$$f(n) \in \Omega(g(n)) \text{ and } g(n) \in \Omega(f(n))$$

Is this statement *always*, *sometimes*, or *never* true? Explain your answer in one sentence.

4. The-ta Knows Best! [6 points]

For each of the following, give a $\Theta(-)$ bound, in terms of n, for the *worst case runtime* of the method.

(a) (2 points)

```java
int hello(int n) {
   if (n == 0) {
      return 0;
   }
   for (int i = 0; i < n; i++) {
      for (int j = 0; j < n * n; j++) {
         System.out.println("HELLO");
      }
   }
   return hello(n - 1);
}
```

Runtime

(b) (2 points)

```java
void whee(int n) {
   for (int i = 1; i < n; i *= 2) {
      for (int j = 1; j < n; j *= 3) {
         System.out.println("WHEE!");
      }
   }
   for (int k = n/2; k < n; k++) {
         System.out.println("WOAH!");
   }
}
```

Runtime

(c) (2 points)

```java
void flipflop(int n, int sum) {
   if (n > 10000) {
      for (int i = 0; i < n * n * n; i++) {
         sum++;
      }
   }
   else {
      for (int i = 0; i < n * n * n * n; i++) {
         sum++;
      }
   }
}
```

Runtime

5. **Analysis** [6 points]
For each of the following, determine the runtime of the algorithm/operation:

Runtime

(a) (2 points) Worst case `find` in a B-Tree

(b) (4 points) Best case `insert` in a hash table with size n and a current $\lambda = 1$ if the collision resolution strategy is:

Runtime

- Separate Chaining

Runtime

- Double Hashing

6. Choose The Data Structure [15 points]
For each of the following, decide which data structure is most appropriate to solve the problem:

AVL Tree	B-Tree	Bit Set	FIFO Queue	Hash Table
Linked List	Heap	Stack	MTF List	Vanilla BST

(a) (3 points) An operating system needs to schedule when to run each of the current processes.

(b) (3 points) You want to store 1MB of *non-comparable* data and you expect to run the `find` operation very frequently.

(c) (3 points) You want to keep track of ASCII characters that are allowed in a valid password.

(d) (3 points) Google (which has petabytes of data) wants to store search queries in such a way that they can easily find the closest alphabetical query to a new user query.

(e) (3 points) A grocery store wants to make a database of their products (they only have about 5000 unique products) so that employees can look up the aisle that a particular product is in by name.

Basic Techniques.

This part will test your ability to apply techniques that have been explicitly identified in lecture and reinforced through sections and homeworks. Remember to show your work and justify your claims.

7. AVL Trees [8 points]
Insert $1, 2, 7, 6, 8, 3, 4, 5$ into an AVL tree *in that order*. You do not have to show your intermediary steps, but no work and a wrong answer will receive no credit.

8. Hashing [15 points]
You know in advance that you will never put more than *six* strings into a particular hash table.

(a) (2 points) If you had a choice between 6 and 7 as your table size, which would you choose? Why?

(b) (3 points) Insert 1, 5, 9, 12, 37, 16 into your hash table using the hash function $h(x) = x$ and *separate chaining*.

(c) (2 points) What is the load factor of your hash table from part (b)?

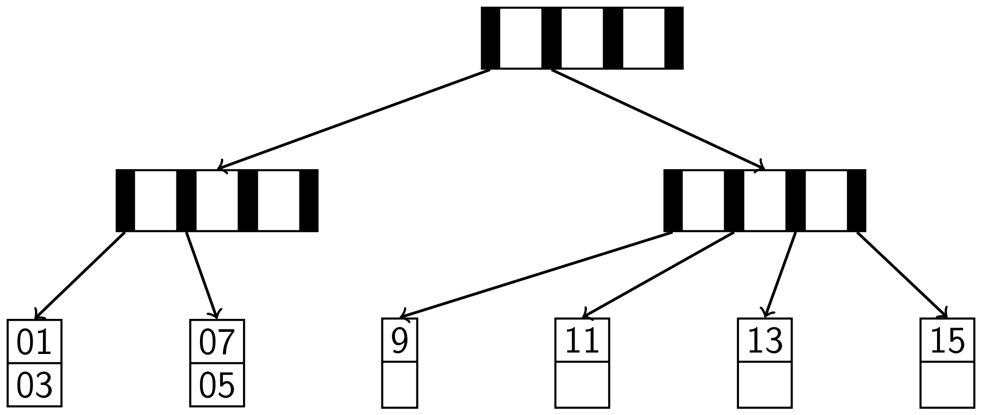
(d) (3 points) Insert the same numbers into an *empty version* of your hash table using the hash function $h(x) = x$ and *quadratic probing*.

(e) (3 points) Does your hash table from part (d) have primary clustering? What about secondary clustering?

(f) (2 points) What is the load factor of your hash table from part (d)?

9. **B-Trees** [10 points]

Consider the following "B-Tree":



(a) (2 points) What are $M =$ $\boxed{}$ and $L =$ $\boxed{}$?

(b) (5 points) List (and fix, if possible) everything that you can find that is wrong with the above B-Tree.

(c) (3 points) Suppose that you know the following facts about a computer system:

- 1 page on disk is 1024 bytes
- Keys are 4 bytes
- Pointers are 8 bytes
- Key/Value Pairs are 32 bytes

If you want to use a B-Tree on this system, what should you choose M and L to be?

A Moment's Thought!

This section tests your ability to think a little bit more insightfully. The approaches necessary to solve these problems may not be immediately obvious. Remember to show your work and justify your claims.

10. Treaps! [15 points]

In this problem, we define and analyze a new data structure called a *Treap*. A *treap* is a combination of a BST (tre-) and a heap (-eap). Operations on treaps are defined as follows:

```
void insert(n)
```

- Generate a *random* priority p.
- Do a standard BST insert into the treap of (n, p) completely ignoring the priority.
- Fix the Treap so that the *keys* follow the *BST Property* and the *priorities* follow the *Min-Heap Property*. (Note that the treap *does not* have to obey the *Heap STRUCTURE Property*.)

(a) (6 points) Suppose that we have a random number generator that will generate the following sequence of random numbers: $5, 3, 7, 2, 4, \ldots$

- Insert c, b, e, q, h into a treap in that order.
  Hint: After doing *each* BST insert, use *AVL Rotations* on the inserted element until the Heap Property is satisfied.

- Is your resulting treap an AVL tree? Why or why not?

(b) (4 points) Suppose we implement `deleteMin` on a treap as follows (where `delete` is defined with lazy deletion):

```
int deleteMin() {
   min = findMinKey()
   return delete(min);
}
```

- What is the best case runtime of this operation?

- What is the worst case runtime of this operation?