CSCI C343 - Data Structures	First Name: _____
Summer 2017	Last Name: _____
Lab 14 Quiz	IU email: _____

Quiz in preparation for the Final Exam

Lab 14 Quiz Due Date and Time

- Friday, June 09 2017, from 12:45PM to 2:00PM.

Work Policy

Work for Lab 14 Quiz is to be completed individually - no group solutions or cooperative work.

Note: this week's Lab 14 is a written quiz, to help you prepare for the Final Exam next week. This quiz needs to be completed at lab-time, in LH030.

Instructions

Please ***read these instructions carefully and completely***.
Ask the person proctoring the test if anything is unclear, or if you have any questions.

The C343 Final Exam (and thus also this practice quiz) must be taken in written form, pen/pencil-and-paper, *not on your computer/tablet/cellphone/etc.* (those have to be all turned off and put away for the entire duration of the test). During the Final Exam, any book or other source, will be off-limits. If you consult any source of information during the Final Exam, at any time before submitting the test, you will receive a score of 0 on the Final Exam, and the incident will be reported to the university.

Note: since this is a practice quiz, you may consult any written notes, your (printed) textbook, etc. while taking the quiz at Lab 14 ; however, that will not be allowed during the Final Exam!

1. You have available 75 minutes to work on the Final Exam (and thus also this practice quiz). You may take a few more minutes if necessary. All submissions have to be turned in by 2:00PM ***at the latest***. Don't wait until the last few minutes. If you finish early, we recommend that you double-check all your answers before turning in your work and leaving the lab.
2. When a question asks to answer with an "expression", it means a mathematical expression, not program code.
3. When a question asks to answer with program code, it does not have to be precisely *exact* Java, as long as the meaning is clear and the sentences are Java-like.
4. For short answer problems, be as clear as possible. When in doubt, give details. Partial credit is possible on many problems, so don't leave anything blank.
5. During the Final Exam, you will not be allowed to consult any books, phones, computers,

calculators, or neighbors. The Final Exam for C343 is closed-book, closed-everything, with one exception: you are allowed ***one*** 3"x5" index card with anything you want personally handwritten on it (both sides). ***No other index card sizes will be allowed, nor cards written by anyone else, printed material (not even on the index card), etc***. Before you leave the lab, write down your name on the index card, and turn it in to the person proctoring the exam.
(for this practice test, you may consult written notes, the textbook, etc -- but no electronic devices nor any internet connection).

6. The Final Exam (and thus also this quiz) is to be turned in before leaving LH030. Late submissions of the Final Exam will not be considered for grading (nor for this quiz).
7. ***Mark*** your answer parts clearly when necessary, e.g. with (a), (b), (c), etc. and please ***write legibly***. Handwriting that is unreadable or very hard to read can not be graded and will receive 0 points.
8. Do ***not*** copy, email, transmit, etc. any part of the exam/quiz (neither questions nor answers). Cheating will be dealt with harshly.

---

1. **Short Questions.**
    1. A hash function maps ***key values*** to ***positions (or indices) in a hash table***.
    2. Two fundamental representations for graphs are ***adjacency matrix*** and ***adjacency list***.
    3. A dynamic programming algorithm that we examined is: ***(several possible answers that are correct...)***
       **** memoized DP algorithm to compute Fibonacci numbers***
       **** bottom-up DP algorithm to compute Fibonacci numbers***
       **** Edit Distance (Levenshtein's DP algorithm)***
       etc.
    4. A DAG is ***a Directed Acyclic Graph***.
    5. Use your own words to describe what is a **hash function**.
       Provide an example (pseudocode) of a hash function.
       A hash function maps key values:
       from a (potentially very large "universe") set of allowed key values
       to a (typically small) set of positions (or indices) in a hash table.
       This is also called a "pre-hashing" function, to be more precise.

```
// example - modular hashing, with hash table of size M:
int h (int x, int M) {
    return x % M;
}
```

    6. What does it mean for two keys to have a collision at a slot in a hash hable?
       Show two distinct examples to demonstrate such a collision.
       Given a hash function h and two keys k_1 and k_2, if $h(k_1) = \beta = h(k_2)$ where β is a slot in the table, then we say that k_1 and k_2 have a collision at slot β under hash function h.
       Two examples are shown at slots 0 and 7 and in Figure 9.3 from the textbook:

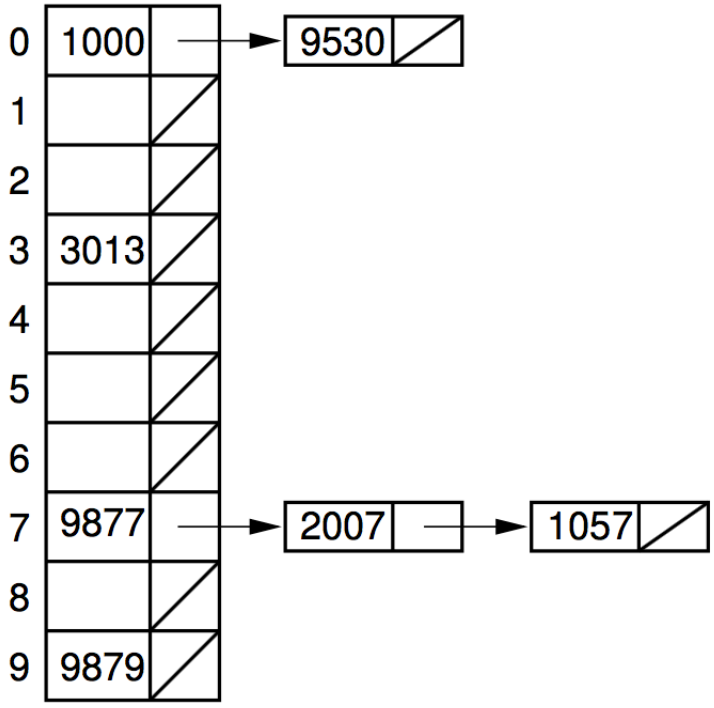
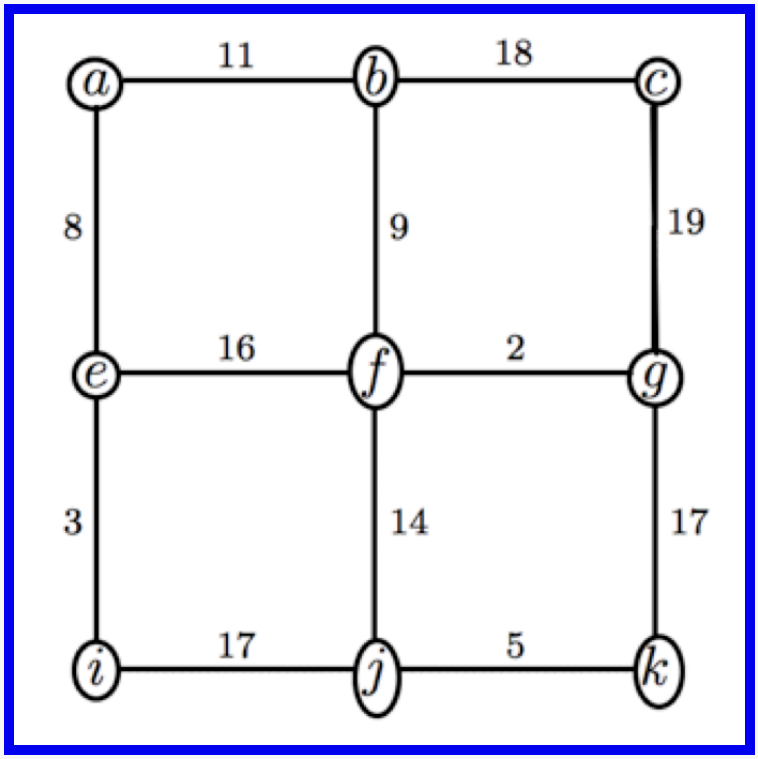


Figure 9.3 An illustration of open hashing for seven numbers stored in a ten-slot hash table using the hash function $\mathbf{h}(K) = K \bmod 10$. The numbers are inserted in the order 9877, 2007, 1000, 9530, 3013, 9879, and 1057. Two of the values hash to slot 0, one value hashes to slot 2, three of the values hash to slot 7, and one value hashes to slot 9.

2. **Graph traversals, BFS, DFS.**

Given the graph shown in the picture:



a) Write down the adjacency matrix representation for the graph.

ANSWER

b) Write down the adjacency list representation for the graph.

ANSWER

c) Show the enumeration of the graph's vertices using the DFS method, starting at vertex ***c***.

ANSWER

d) Show the enumeration of the graph's vertices using the BFS method, starting at vertex ***c***.

ANSWER

e) How many connected components does this graph have? Why?

ANSWER

3. **Hash Tables.**

A `set` is defined as a collection of elements in which any value may occur at most once, and the values are not ordered. Suppose you're given the task to implement the given ADT for `set` as listed here:

```
interface Set<E> {
    boolean contains (E element) ;
    void add (E element) ;
    void remove (E element) ;
}
```

(you don't need to implement it for this quiz: however, answer the following questions about such an implementation)
a) Would it be appropriate to implement `set` using a hash table? Motivate your answer.
yes, a hash table would provide a correct implementation, because there are no key duplicates in hash tables.
And a hash table would also provide an efficient implementation: $O(1)$ ***average case for `contains()/add()()/remove()`.***

b) Assuming a hash table implementation for `set`. To provide a method that returns all elements in a set instance, in any order, what would be the running time? Motivate your answer.
$O(n)$ ***for*** n ***elements, because it takes*** $O(1)$ ***for the `contains()` function on average.***

c) What data structure should be used to implement a modified version of the `set` that also allows order-based operations, e.g. to provide all elements in a set instance? Motivate your answer.
AVL tree, correct because it allows ordering (previous/successor) and efficient because it provides $O(logn)$ ***asymptotic run time.***

Lab 14 Submission Instructions:

- Turn in your written quiz to the instructor by the end of lab time: Friday, June 09 2017, 2:00pm.

Last updated: June 09, 2017
mitja⊛indiana·edu