CSE 373 Lecture 14 In-Class Worksheet — Fall 2018

Name: _____ Date: _____

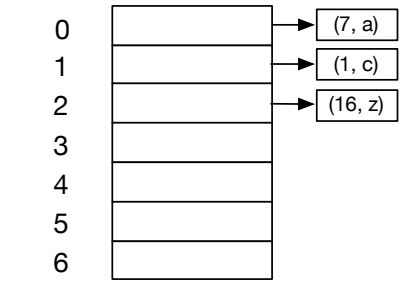
UW Student No: _____ **UW Email address**: _____

Partner name(s) for this activity: _____

Will you want to pick up your worksheet later? Circle one: Yes / No

(Q1) Each of the following table uses the hash function h(x) = x % 7, but different collision handling strategies (described above the table). Insert the key-value pairs (8, a) and (14, b) in each table.

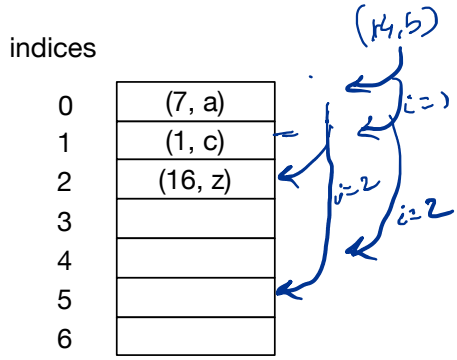
Separate chaining



Open addressing with linear probing

indices	
0	(7, a)
1	(1, c)
2	(16, z)
3	
4	
5	
6	

Open addressing with quadratic probing



(Q2) Say we need to store records of CSE 373 students in a hash table for quick look up. A student record is a string with comma separate values, and contains following info:

- Student number (integer),
- UW Net ID (string),
- First name (string),
- Last name (string),
- Section (string),
- Class (string),
- Major (string),
- UW Email (string)

Example of a student record

value = “16124187,alice12,Alice,Smith,AG,JUNIOR,Electrical Engineering,alice12@uw.edu"

(2a) Our task is to come up with a key for each record so that we can store (key, value) pair in a hash table. The key can be a string or an integer. Suggest which fields you'll use to create a key. Justify your answer.

(2b) Suppose you are told that the key for each student record is the student's UW email, and your task is to create a hash function that translates UW email to a non-negative integer. Are there parts of the key that you'll want to discard before you apply your hash function? (e.g., discard first two characters, discard last two characters). Explain what you'll discard and why.

(Q3) You are given an array of size 10 to use for your hash table. You are not allowed to change the array size. The keys in the (key, value) pairs that will be stored in the hash table are always in multiples of 15, i.e.., the keys are from this sequence 15, 30, 45, 60, …. The table uses a simple mod as its hash function, h(k) = k % x.
What are some good choices for x (select all that apply)?:

a. 10 — too many collisions (all keys would hash to 0 or 5)
b. 15
c. 7 — fewer collisions but wasted table space
d. 9 — index 8, 9 or 9 won't be used
e. 11

(Q4) Should the hash function always return a non-negative integer that is less than the table size? What should we do if the hash function returns an integer greater than our table size?

mod it by table size.

So, array index = hashed key % array length

(Q5) What are some expected requirements (or properties) for a hash function? Are any of these strict requirements for a hash function?

1. Determinism ⟹ $h(K_1) = h(K_2)$ if $K_1 == K_2$ — strict requirement
2. Quick
3. Uniformity

→ soft requirements