You're expected to work on the problems before coming to the lab. Discussion session is not meant to be a lecture. TAs will guide the discussion and correct your solutions if needed. We may not release solutions for all problems. If you're better prepared for discussion, you will learn more. TAs will record names of the students who actively engage in discussion and report them to the instructor. The instructor will factor in participation in final grade.

1. (Basic) What is the main advantage of hash tables over direct-address tables?
   Sol. It uses much less space. If the universe of possible keys is super large compared to the set of actual keys, then lots of space is wasted.

2. (basic) Consider a hash table with $m = 11$ slots and using the hash function $h(k) = k \bmod 11$. Say we insert elements $k = 41, 18, 3, 8, 19, 5, 1, 7$ in this order. Show the final table when using chaining to resolve collisions. Please insert the element at the beginning of the linked list. **Sol.**

   Pos 0:
   Pos 1: 1 →
   Pos 2:
   Pos 3: 3→
   Pos 4:
   Pos 5: 5→
   Pos 6:
   Pos 7: 7 → 18→
   Pos 8: 19→ 8 → 41→
   Pos 9:
   Pos 10:

3. (basic) What is a universal hash family? **Sol.** See the lecture slides.

4. (intermediate) Suppose we choose from between the function of x hashing to the slot corresponding to the last $\log_2 n$ bits of x and that of x hashing to the slot corresponding the first $\log_2 n$ bits of x? Is this a universal hash family? **Sol.** No. Two keys with the same first $\log_2 n$ and last $\log_2$ bits will hash to the same slot, no matter which hash function is chosen from the two.

5. (Intermediate) We are given two arrays $A[1 \ldots m]$ and $B[1 \ldots n]$. We want to test if the set of numbers in $B[1 \ldots n]$ is a subset of numbers found in $A[1 \ldots m]$. For simplicity, assume that all numbers in each array are distinct. Give an algorithm that has a running time $O(m + n)$ in expectation. **Sol.** Insert all numbers in A into a hash table. Then, we just need to search each element in B in the hash table. Since each operation takes $O(1)$ time in expectation, all take $O(m + n)$ in expectation.

6. (Intermediate) Suppose we are given two linked lists A and B which includes m elements and n elements, respectively. Let's think of each list as a set of elements appearing in the list. For simplicity, assume that all elements in each list are distinct. Give an algorithm that returns the intersection of the two sets in $O(m + n)$ time in expectation. **Sol.** Omitted.

7. (Advanced) Suppose we are given an array $A[1 \ldots n]$ of integers, and another integer $\Delta > 0$. We want to know if there exist $1 \leq i, j \leq n$ such that $|A[i] - A[j]| = \Delta$ in $O(n)$ time in expectation. Describe an algorithm for that. **Sol.** Sketch: What we want know is the two

sets $\{A[1], A[2], ..., A[n]\}$ and $\{A[1] + \Delta, A[2] + \Delta, ... + A[n] + \Delta\}$ have a common integer, which can be done in $O(n)$ time using hashing.

8. (Intermediate) You are given two sequences, $\langle a_1, a_2, ..., a_n \rangle$ and $\langle b_1, b_2, ..., b_n \rangle$, where each sequence consists of distinct integers. Describe a linear time algorithm (in the average case) that tests if a sequence is a permutation of the other. Assume that the simple uniform hashing assumption holds. Explain the running time of your algorithm.
   Sol. We first insert $a_1, a_2, ..., a_n$ into the hash table of size $\Theta(n)$. This is done in $O(n)$ time in expectation. Then, we search each b_i in the hash table, which is done in $O(1)$ time. If we can find every b_i in the hash table, it, together with the fact that each of the two sequences consists of distinct integers, it means that one is a permutation of the other. The (expected) running time is clearly $O(n)$.

9. (advanced) Consider again the previous problem but now allowing both the input sequences to have repeated integers. Modify your algorithm so that it also works in this case.