### Instructions.

1. You could submit a pdf of the homework, either printed or hand-written and scanned, as long as it is **easily** readable.

2. If your solution is not clearly written, it might not be graded.

3. Prove the correctness of your answer. A correct answer without a proof might not be accepted.

4. If you have discussed the solution with other students, mention their names clearly on the homework. These discussions are not forbidden and are actually **encouraged**. However, you must write your whole solution yourself.

5. All questions have same weight.

6. You could refer to a data structure studied in class, and just mention briefly their guarantees. For example “**It is known that a Red-Black tree could support the insert/delete/find operations on a set of** \( n \) **elements in time** \( O(\log n) \).”

7. If your answer uses one of the data structures or algorithms that were studied in class, you could refer to it without having to repeat details studied in class. If your answer requires only modifications of one of the algorithm, it is enough to mention the required modifications, and what’s the effect (if any) on the running time and on other operations that the algorithm performs.

8. In general, a complete answer should contain the following parts:

   (a) High level description of the data structures (if needed). *E.g. We use a binary balanced search tree. Each node contains, a key and pointers to its children. We augment the tree so each node also contains a field...*

   (b) High level description of the algorithms

   (c) Proof of correctness (why your algorithm provides what is required).

   (d) A claim about the running time, and a proof showing this claim.

9. When discussing trees, do not confuse the *children* of a node with the *descendants* of the node.
1. Assume a hash table $T[0..15]$ (that is, $m = 16$), and a open addressing hashing where
\[ h(x, i) = \{ x + i \cdot (x \mod 10) \} \mod m. \]

Assume you start with an empty table. Show an example of a set of three distinct keys \( \{k_1, k_2, k_3\} \) such that
(a) You could not insert all of them into the table, and
(b) $k_j \mod 10 > 0$ for $j = 1, 2, 3$.

2. Under the same assumptions as the previous question, but now pick $m = 17$ (rather than $m = 16$). Prove that there is no such set of 3 different keys.

3. Under the same assumptions as the previous question, but now pick $m$ to be any prime $\geq 17$. Prove that there is no such set of 3 different keys. You could use the following known result. Assume $m$ is a prime. Let $a, j_1, j_2$ be three integers, all $\geq 1$ and $\leq m - 1$. Assume $j_1 \neq j_2$. Then $(aj_1) \mod m \neq (aj_2) \mod m$.

4. Under the same notations as the slides. Assume $h(x)$ is any hash function, mapping keys from a universe $U$ onto a table $T[0..m - 1]$. Show that if $|U| \geq m^2$ then there is a set $K \subset U$ such that all keys of $K$ are mapped by $h(x)$ into the same slot of $T$ and $|K| \geq m$. That is, every pair of keys from $K$ creates a collision.

5. Discuss pros and cons of Open Addressing Hashing (where collisions are revolved using double probing) vs. Chain Hashing.

6. Explain how you would use hash functions to find if your computer contains two identical image files (possibly under different names). Give a pseudocode of your solution. Specify which and how your hash functions are used. Do not use values provided by the file system.

7. Repeat, but this time you could use values provided by the File System/Operating System (such as MD5).

8. Assume you start with a hash table of size 1024, and which currently contains 500 keys. You need to insert new keys as they arrive, and you are not sure how many keys are about to be inserted, so the number of keys is not known in advance. No deletions are needed. Explain how you would expand the table, such that the load factor $\alpha$ is always between 0.333 and 0.5, and the total expected time for inserting any sequence of $n$ keys is $O(n)$. Show a pseudo code of your insertion algorithm, and prove your result.

9. Consider a open addressing function $h(k, i) = (h_1(k) + i) \mod m$ that maps keys into a hash table $T[0..m - 1]$. Here $h_1(k) = (k \cdot B) \mod m$, and $B$ is any fixed integer.

You have inserted the set of keys $j \cdot m$ (for $j = 0, 1, 2 \ldots \left\lfloor \frac{m}{2} \right\rfloor$). Next you are about to insert another single key $k$. 
Assume that the key $k$ could be mapped into each one of the $m$ slots with equal probability. What is the expected time that this operation would take?

If you are not familiar with computing expectation, just discuss the possible cases for the time needed to insert $k$ (the answer is different, depending on what is the value of $h_1(k)$).