Problem set 9.

2-3 Trees, Hash functions.

1. Insert the following elements in the order given, into an empty 2-3 tree $21, 8, 63, 5, 69, 32, 7, 23, 25$ in it. Then delete, in order, $7, 25, 21, 5$.

2. We want to store items in a hash table using the hash function $h(k) = k \pmod{11}$, and we will resolve collisions with chaining.
   (a) Insert the following elements into an empty hash table: $2, 5, 12, 1, 3, 88, 23, 43, 10, 34$.
   (b) Demonstrate how SEARCH(1), SEARCH(20) and SEARCH(45) will work in the resulting table.
   (c) Delete the element 23.

3. We want to store items in an open-address hash table using the hash function $h(k) = k \pmod{11}$, with collisions resolved using a linear probe.
   (a) Insert the following elements: $6, 5, 11, 17, 16, 3, 2, 14$.
   (b) In the resulting table, search for the element 16 and show the steps the algorithm will take.
   (c) Delete 2 from the table and search for 15 in the resulting table.
   (d) Insert 25 into the resulting table and report its address.

4. We want to store items in an open-address hash table using the hash function $h(k) = k \pmod{11}$. After the insertion of $4, 5, 14, 15, 16, 26, 3$, what will the resulting table look like and what are the number of collisions at each insert if collisions were resolved using:
   (a) Linear probe (b) Quadratic probe
   (c) Double hashing where our second hash function is $h'(x) = 7x \pmod{(10)} + 1$?

---

5. You are given a 2-3 tree such that the root contains three subtrees and the two nodes stored in the root are 40 and 50. What is the minimum and the maximum number of elements contained in the tree, if you know that the tree only contains positive integers?

6. Given an open-address hash table of length 11 with hash function $h(x) = x \pmod{11}$ and using linear probe for collision resolution, we perform a few insertions and deletions and obtain the following table ($*$ denotes a deleted entry and not an empty one):

0	1	2	3	4	5	6	7	8	9	10
11	4	*	3	15		6	18			22

   There is only one element in the above table that upon deletion and being inserted again, will end up with a new address in the hash table. What is this element? Show the deletion and insertion steps of it to support your answer.

7. A 2-3 tree contains all integers in the interval $[1, 178]$. The root is a 3-node and one of the elements it contains is 17. What can be the other one?

8. In table $T[0 : M]$ we have placed $2n$ elements ($n < M/3$) using some hash function, when we found that the elements were all placed in one of the first $3n$ positions. If there was no deletion in the meantime and at the end of the table all places with index $3i$ remained empty ($0 \leq i < n$), then there could be at most how many collisions if
   (a) a linear probe was used?
   (b) a quadratic probe was used?

9. A hash table of size m already has some elements, but the table is not full. Give an algorithm with running time $O(m)$ that determines the maximum number of collisions that can happen during the insertion of a new element using a linear probe to resolve collisions.

---

10. Can it happen that in an open-address hash table of size $n > 3$ with exactly 3 elements in it, but a search takes $O(n)$ steps?

11. For a hash table of size $M = 7$, is the following a good hash function: $h(x) = x^2 \pmod 7$?

12. Odd integers between 0 and 70 are inserted, in some order, into an empty open-address hash table of size 47, using linear probe for collision resolution. The hash function is: $h(x) = x \pmod{47}$. Show that no matter what order the numbers arrive in, the number of collisions is the same for all orders.

13. All integers between 1 and 91 divisible by 3 are inserted into an empty open-address hash table of size M, with hash function $h(x) = x \pmod M$, and using a linear probe for collision resolution. How many collisions could occur if $M = 35$? If $M = 36$?

14. We want to store 0-1 bit sequences of length $n+1$, of the form $b_0 \ldots b_n$. We say that the bit b_0 determines the parity (odd/even) of these sequences when considered as a binary number. If we use open-address hash table with $h(x) \equiv x \pmod M$ as the hash function and using linear probe for collision resolution, determine which of the two possible values of M, $M = 2^n$ or $M = 2^n + 1$, will give fewer collisions.