1. (a) We wish to hash approximately 30,000 keys using the division method and chaining. What would be a suitable size for the hash table, if we want to have a load factor of at most \( \frac{3}{m} \)?

(b) Again, we wish to hash approximately 30,000 keys and use chaining with the load factor at most 3. This time we use the multiplication method, and assume the keys are 64-bit non-negative integers. What would be good values for \( A \) and \( m \)?

2. The keys 67, 238, 105, 189, 52, 244, 132 and 301 are inserted into a hash table with chaining. The size of the table is \( m = 7 \).

(a) Show the final state of the data structure, when hashing is done by the division method.

(b) Show the final state of the data structure, when hashing is done by the multiplication method with \( A = 0.62 \).

(Notice that the parameter values here are unrealistic to keep the calculations simpler.)

3. Suppose we wish to use an extremely large hash table. Due to the size of the table, we do not want to initially go through the whole table and set the values to \texttt{NIL}. However, we need some way of telling, whether the value we find in location \( A[h(k)] \) is a real key that has been inserted in the table or some random garbage that was left in the storage when the table was allocated.

Design a data structure in which

- initialisation takes only a constant time
- operations \texttt{SEARCH} and \texttt{INSERT} take only a constant time
- the space requirement in excess to a normal hash table is \( O(n) \), where \( n \) is the number of inserted keys.

For simplicity, assume that there are no collisions, \( n \) is known in advance and we do not need to implement \texttt{DELETE}.

\textit{Hint:} Maintain a stack that contains the keys that have been inserted into the table. The actual table contains links to this stack.