7. **Variations on a theme.**

Another standard method for handling collisions in hash tables is to overflow colliding items into a smaller *global* secondary hash table. If we implement the secondary table recursively, we obtain an logarithmic sequence of hash tables, each a constant factor smaller than its parent. By making the table sizes successive powers of two, we can maintain the entire sequence in a single array.

Let $m = 2^r$ for some integer r. Our hash table consists of a single array $T[1..2m-1]$, which is notionally divided into subtables of the form $T[2^i..2^{i+1}-1]$ for each integer $0 \le i \le r$. For each index i, we use a separate hash function $h_i \colon \mathcal{U} \to [2^i]$. Here are the algorithms to insert and find an item x.

```
INSERT(x):
  for i ← r down to 0
      h ← h_i(x) + 2^i
      if T[h] = x
          return TRUE
      else if T[h] = ∅
          T[h] ← x
          return TRUE
  return FALSE
```

```
FIND(x):
  for i ← r down to 0
      h ← h_i(x) + 2^i
      if T[h] = x
          return TRUE
      else if T[h] = ∅
          return FALSE
  return FALSE
```

(a) Assume that $h_0, h_1, \ldots, h_r$ are fully independent ideal random functions. Suppose we INSERT $n = m/2$ distinct items into an initially empty table.

- Give the best bound you can on the probability that all n calls to INSERT are successful. *[Hint: What is the expected number of collisions at the top level?]*
- Assuming all n insertions were successful, prove that the expected running time of FIND(x) is $O(1)$.

(b) Repeat the previous analysis, but with $n = m$. *[Hint: What is the expected number of* ***empty cells*** *at the top level?]*

(c) Now suppose each hash function h_i is merely 2-universal, but the different h_i's are mutually independent from each other. Repeat your analysis from part (a), assuming $n = m/2$.

*(d) How large can you make n and still derive an $O(1)$ expected time bound for FIND(x) with only 2-universal hash functions? Alternatively, how much independence is required to guarantee $O(1)$ expected time bound for FIND(x) when $n = m$?

*(e) Now suppose instead of using an independent hash function at every level, we use a *single* hash function $h \colon \mathcal{U} \to [m]$, and implicitly define $h_i(x) = \lfloor h(i)/2^i \rfloor$.

Equivalently, we can model our data structure is a complete binary tree with depth r. The INSERT algorithm chooses a random leaf, and then walks upward toward the root to find an empty node to store x. In the implementation below, the overflow tree is encoded into a single array T in heap order.

```
INSERT(x):
  h ← h(x) + 2^r
  for i ← r down to 0
      if T[h] = x
          return TRUE
      else if T[h] = ∅
          T[h] ← x
          return TRUE
      h ← ⌊h/2⌋
  return FALSE
```

```
FIND(x):
  h ← h(x) + 2^r
  for i ← r down to 0
      if T[h] = x
          return TRUE
      else if T[h] = ∅
          return FALSE
      h ← ⌊h/2⌋
  return FALSE
```

Repeat your analysis from part (a), assuming $n = m/2$ and the hash function h is ideal random.

*(f) Prove something interesting about the previous hashing scheme when $n = m$ and/or when the hash function h has limited independence.