READ THIS NOW!

- Print your name in the space provided below.
- There are 3 short-answer questions, priced as marked. The maximum score is 50.
- Most questions require analyzing a scenario, drawing a conclusion, and justifying that conclusion. Credit will be awarded for making statements that are both true and relevant to the question. Do not feel obligated to fill all the available space when answering the questions.
- This examination is closed book and closed notes.
- No calculators, cell phones, or other computing devices may be used. The use of any such device will be interpreted as an indication that you are finished with the test and your test form will be collected immediately.
- Until solutions are posted, or gone over in class, you may not discuss this examination with any student who has not taken it.
- Failure to adhere to any of these restrictions is an Honor Code violation.
- When you have finished, sign the pledge at the bottom of this page and turn in the test.

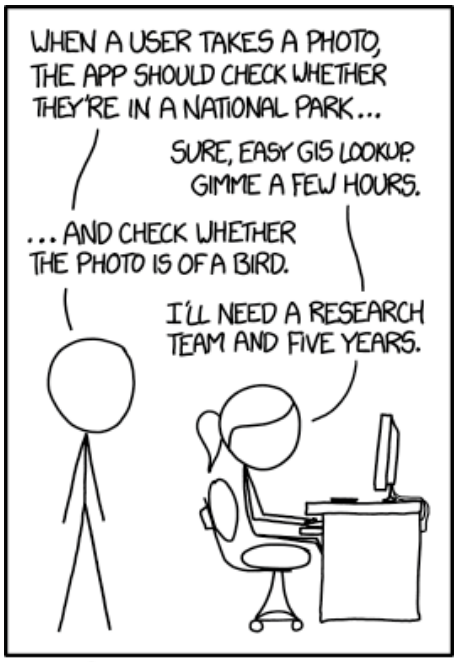
Name (Last, First) _____

printed

Pledge: On my honor, I have neither given nor received unauthorized aid on this examination.

_____

signed



In the 60s, Marvin Minsky assigned a couple of undergrads to spend the summer programming a computer to use a camera to identify objects in a scene.

He figured they'd have the problem solved by the end of the summer.

Half a century later, we're still working on it

xkcd.com

1. Suppose that we insert 150 elements into a previously empty hash table, using an array of dimension 300, and that the hash function maps exactly 9 of those 150 records to slot 42 of the table.

a) **[10 points]** Suppose that the hash table resolves collisions by linear probing. If we now perform a search for one of those 9 records, what is the maximum number of record comparisons that might be required? Explain.

150.

How? We don't know many collisions occur during the insertions, but it is possible that every element, except the first one, collides with another element that was previously inserted. In that case, it's possible that the only element that's actually in its home slot is the first one that was inserted.

b) **[10 points]** Suppose that the hash table resolves collisions by using a chain, consisting of a singly-linked list, in each tables slot. If we now perform a search for one of those 9 records, what is the maximum number of record comparisons that might be required? Explain.

9.

We know that those 9 records are the only ones that hash directly to slot 42, and since we're using chaining, no other records get put into slot 42 by probing, so slot 42 will contain exactly those 9 records.

---

2. **[10 points]** A designer working on a hash table decides that, for each record inserted to the table, she will also store the value the hash function computed when applied to the key for that record. The designer explains her decision with the somewhat cryptic comment "it will make some operation(s) more efficient".

What operation(s) might she have in mind, and why this would improve their efficiency. Or, is she incorrect, and this is merely a waste of memory? Either way, explain.

This would make resizing the hash table more efficient.

In order to resize the table, we must recompute all the home slots, and that requires modding the hash value for each record by the new table size.

Evaluating the hash function is likely to be quite a bit more expensive than modding, so if we saved the hash value for each record, we would avoid that cost.

3. Suppose that we are designing a hash table to store thousands of GIS records, using the feature name as the key. For simplicity, we will assume that any duplicate feature names have been eliminated before we hash the keys. We have decided on a table size that is about 1.5 times the number of records that wind up being inserted to the table. We have also chosen a reputable hash function for handling keys which are strings. We have also settled on using some form of quadratic probing to resolve collisions.

Unfortunately, in testing we discover that the number of primary collisions is unacceptable. (A *primary collision* occurs when we get the same home slot index by modding the hash value by the table size.)

Which of the following ideas, if any, has the potential to reduce the number of primary collisions? Explain.

a) [10 points] Changing the collision resolution strategy.

No.

The collision resolution strategy doesn't even come into play until a collision has occurred.

b) [10 points] Choosing a larger table size.

Yes. Let's say the old table size was N, and the new table size is M.

A primary collision would have occurred in the old table when we have two records, say A and B, such that H(A) % N == H(B) % N.

Now, if H(A) == H(B), then modding by a different table size will make no difference; A and B will still collide (although probably in a different slot than before).

On the other hand, if H(A) != H(B), then it's likely A will now map to a different home slot than B, avoiding a primary collision.