	Name: Trevor Buck
	ID: 109081318
CSCI 3104, Algorithms	**Profs. Grochow & Layer**
Explain-It-Back 4	**Spring 2019, CU-Boulder**

The ecology department is planning a large-scale fish migration study that involves electronically tagging and releasing millions of fish across North America, waiting six months, then trapping the fish and recording the where they found each fish, according to its tracking number. In previous smaller-scale experiments, the field scientists used a hand-held device that had a sensor for reading the electronic sensor and a small onboard hard drive that used a predefined table for storing the tag ID, timestamp, and current GPS. In this table, every possible tag had a preset row which allowed for very fast (constant-time) insertions and lookups. While the team would like to re-use this hardware, they do not think that there is enough hard drive space to account for a table with millions of rows. Help them figure out another solution that provides fast insertions and lookups without requiring large memory allocations. HINT: an individual scientist will only tag a few thousand fish at a time.

To whom it may concern in the Ecology Department,

I heard from someone in your department that you had concerns about storing all your data from your new experiments with fish migrating. They mentioned that your hard drive wouldn't have enough space to account for a table with millions of rows. That is correct. But there is good news! By using some fancy memory storage tricks, we can make sure that your current hardware holds all the information in an efficient and effective manner.

I want to talk to you about two specific memory management strategies that could prove useful in this situation. The first is called a hash-table, and the second is a linked-list. A hash function is a special type of table. From what I heard from your colleague, it sounds to me that you understand what a table is. Each fish is given a specific 'box' or 'spot' in the table and that is where all the information pertaining to that particular fish is stored. In this implementation, your table has a ratio of one-to-one. One box per one fish. A hash table differs in that it has a *many-to-one* ratio. One box contains many fish. This is simple in concept but presents a couple of challenges. How do we sort the fish into the boxes? How do we know where the fish's information is, after it's been sorted into the box? Assuming we find the right box, how do we sort through all the information in the box to find the right fish? Luckily for us, all these questions have answers.

As far as sorting the large amount of fish into the smaller amount of boxes, we would use something called a hash function. As you know, a function takes an input, and produces an output. In our example, it takes a fish's ID number, and produces a 'box' number (I.e., Somewhere in our hash-table). A common way to do this is by dividing the ID number by the amount of boxes, and then outputting the remainder. For example, if you had 1024 boxes in your hash-table, fish with the ID of 1 or 1,025 or 2,049 or and 10,241 would all be in the same box. This is useful, because as long as you know the hash function, and the fish's original ID, you can *always* find where they would be stored in the hash table.

Now assuming we can get to the right box, how do we sort through all the fishes in that box? Well do you remember when I mentioned a linked-list? Because this is where those come in to play. A linked list is a lot like what the name says. It's a list of items that are linked. ☺ What I mean by that is each fish has a 'pointer' to the next fish in the box. Basically, it says, "If I'm not the fish you are looking for, check the next one. Here is where you can find the next one. Its' address is: ______"

To sum it all up, I'd like to use one last piece of imagery. Let's pretend that you have a reallby big filing cabinet with lots of drawers. And in each one of those drawers, there are folders. In this scenario, each drawer would represent a different 'box' in our hash-table. Each folder, represents a single item in our linked list. In order to get information about a specific topic or fish, you would first have to find the right cabinet using a hash function. You would then look at each folder in that drawer, one at a time, until you found the right folder. Finally you could open that folder and find all the information that you were looking for.

Anyway, I hope this helps! Please contact me if you have any questions, or if you would like any help implementing these ideas in your current hardware.

Best of luck with your fish migration studies!!

Sincerely,

Trevor Buck