DSC 40B - Hashing

Problem 1.

Suppose a hash table with 1000 bins stores 2000 numbers. Collisions are resolved with chaining.

True or False: it is possible for one of the hash table's bins to contain zero elements.

Problem 2.

Suppose a hash table is implemented so that it has ten bins, and that this number of bins is not increased when new elements are inserted; that is, the hash table is not allowed to ”grow” or ”resize”. Let n be the number of elements stored in the hash table.

What is the expected time complexity of querying an element in this hash table, as a function of n? You may assume that collisions are resolved with chaining, that the bins are linked lists, and that the hash function is ”good” in that it appears to uniformly distribution elements among bins.

Problem 3.

What is the expected time complexity of the code below, assuming that *numbers* is a Python *set* of size n? Recall that Python's sets are implemented as hash tables.

```python
def foo(numbers):
    count = 0
    for x in numbers:
        if -x in numbers:
            count += 1
    return count
```

Problem 4.

Recall that a *mode* of a collection is an element which occurs with the greatest frequency. For example, 4 is a mode of the collection $4, 5, 8, 3, 4, 2, 4, 5, 5, -2$. 5 is also a mode, since it occurs just as frequently as 4.

Describe an algorithm that finds the mode of a collection of numbers.

Problem 5.

Given two strings `a` and `b`, determine if they are isomorphic.

Two strings `a` and `b` are isomorphic if the characters in `a` can be replaced to get `b`. They must also have the same length.

Eg. ”foo” and ”bar” are not isomorphic, but ”egg” and ”add” are. ”badc” and ”baba” are not isomorphic.

Describe an algorithm that determines if two strings are isomorphic.

Essentially, can the characters in `a` be mapped to the characters in `b` such that the characters in `a` can be replaced to get b.