Name: ____

Section: ____

Today's Goals:

- (TBA)

Today's Question(s)

(TBA)

QOTD: Write the corresponding code: `// bessie is a pointer to a Cow whose value is the location on the heap of a new default Cow.`

Key Terms: Hash table, hash function, perfect hash, collision resolution, separate chaining, load factor

Class Exercise

Imagine the following scenarios:

- Implementing a spell-checker (Is word w in the English language?)
- Implementing a dictionary (What is the definition of word w?)

What operations should these data structures support?

- What data structures might you use?
- If you had the first, how would you use it to implement the second?

Associative Containers

- Sets:
- Maps:

Exercise: Comparing Data Structures

For each, what is the complexity of lookup/insert?

- Linked List
- Binary Search Tree
- Array

Hash Functions

- We start with a fixed-size array (say size 10,000)
- Everything we'll insert has a designated place to go in the array.

```
void insert(const Item & key) {
    // ...
    // if this key's spot in the array is empty
    // put key in this key's spot
    // ...
}

iterator find(const Item & key) {
    // ...
    // if this key's spot in the array is empty
    // return end
    // else
    // return iterator to key in spot
}
```

Example Hash Values

```
int computeHash(const string & key) {
    const int sum = sumOfLetters(key);
    return ((sum % 39) + (sum % 25) + (sum % 5));
}
```

key	sumOfLetters	hash	index
bacon	35	45	5
beans	41	19	9
cheese	45	26	6
ice cream	57	27	7
sausage	73	60	0
steak	56	24	4
tea	26	28	8

Perfect Hash Functions

A perfect hash function

-
-

Writing a perfect hash function is not easy.

Alternative:

Imperfect Hash Functions

Collisions will Happen!

Evaluating Hash Functions

What do you think of these string hashes?

```
int computeHash(const string & key) {
    return key[0];

    if (key[0] == ‘a’) return 17;
    if (key[0] == ‘b’) return 495;

    return key.length();
    return sumOfLetters(key);
    return productOfLetters(key);
    return random number;
```

Class Exercise: Hashing