CS 223

November 6th Quiz Notes

Given Study Questions:

1. Given several B-trees, insert this item.

2. Write pseudocode for k-d tree insertion (recursively)

3. Write pseudocode for B-tree depth first search

Assuming that the `B-tree` has a structure like this:

```
Bnode:
    int k_array[order-1]
    Bnode* child_array[order]

search(Bnode* node, int sVal){
    for (int i = 0; i < (order-2); i++ ){
        if (node.k_array[i] == sVal ){ // WE FOUND IT!
            std::cout << node.k_array[i] << std::endl;
        } else if ( node->k_array[i] > sVal){

// The key is larger, so we know our value is smaller than the
// key, so we recursively call search on the smaller element.

            search(node->child_array[i], sVal);
        } else {
            search(node->child_array[order-1]);
        }
    }
}
```

4. Write hash function to store an element in a table of `HT` size

So this is pretty easy, *I think*.

```
// Assuming our value is an integer, and our hash table is
// just a series of pointers to integers

store_val_in_table(int ourVal, int* HT, int tSize){
    HT[ (ourVal%tSize) ] = ourVal;
```

```
    return HT;
}
```

5. Define 3 ways to resolve hash collisions

1. Probing
    - Quadratic
        - Use math to calculate an offset to check for open space
    - Linear
        - Where you look iteratively forward, checking for open space
2. Separate chaining
    - Each index in the table is actually a separate data structure like an AVL tree or something
3. Double hashing
    - Where you hash the result of the previous hash to find a new index to store it in

6. Using STL Map, count # of times you see x in a list (vector)

```
// Assuming that our map is instantiated with both keys and
// values as integers, e.g. ‘map<int, int> myMap‘

for (int i = 0; i < myVect.size(); i++){
    if (myVect[i] == x){
        if (myMap[x]) {
            myMap[x]++;
        } else {
            myMap[x] = 1;
        }
    }
}
```

7. Given a heap, insert x, delete x

- In book Ch. 6