Final Review

hash_map<student, set<class>> studentClass;

There are S students and an average of C classes per student.

What is the big-O of:

- printing all sutdents alphabetically, and for each student listing each of their classes alphabetically
  - S•logS + S•C
- determining who’s taking CS32
  - S•logC
- determining if Joe Smith is taking CS32
  - logC
- determining if anyone took a course with a 2 in the course name
  - S•C

Given a randomly ordered array of N integers:

What is the big-O of the most efficient algorithm for determining if any single number in the array makes up more than 50% of the array

- Approach 1: Use a secondary data structure
  - Create an unordered_map mapping each number in the array to a count and then check the counts
  - **O(N•logN)**
- **Appraoch 2: Use mergesort**
  - **The middle item will be the number that makes up more than 50% (if one is guaranteed to do so)**
  - **O(N•logN)**
- **Approach 3: Quicksort partition**
  - **O(N)**

Given a binary search tree:

What is the algorithm to print it out in reverse order? What is the big-O?

- **In-order traversal but visit the right before the left**

```
void printRev(Node* p) {
    if (p == nullptr) return;
    printRev(p->right);
    cout << p->val;
    printRev(p->left);
}
```

Insert pre-order traversal of tree into hash table of size 7

10 7 3 3 12 11 14
mod
3 0 3 3 5 4 0

0	7
1	11
2	14
3	10
4	3
5	3
6	12

#5 What is the big-O of the following algorithm?
N^2 • log(N^2) = N^2 * logN

#7

`6 16 3 19 13 72 13 12 99`
-> efficient heap-sort on it

6
16 3
19 13 72 13
12 99

// Swap with bigger
6
16 3
99 13 72 13
12 **19**

// Go up and swap with bigger
99
19 72
16 13 3 13
12 **6**

99 19 72 16 13 3 13 12 6

Remove an element
72
19 13

16 13 3 6
12

72 19 13 16 13 3 6 12 99