The exam will cover all material that we covered in class through March 15. The major topic categories after those are:

- Chapter 1 lecture notes (excluding for proofs): §1.1, §2, §3, §4
- Chapter 2 lecture notes: §1, §2, §3
- All of Chapter 4, including general trees, binary search trees, AVL trees, and B-trees
- Hashing and hash tables up to and including linear probing

For each topic, performance analysis is part of the topic. For example, the running times of the various insertion and deletion algorithms, or results concerning the sizes or heights of trees, are questions that could be on the exam.

The format of the exam includes true/false questions, short answer questions, and questions that ask you to analyse algorithm performance, carry out algorithms on examples, or write small chunks of code.

Some sample questions of various types with some solutions are below. No explanations are given. This is just for you to check your work, nothing more.

1. True or False? If a function's worst case running time is $O(n^3)$, it can never run in strictly linear time. **FALSE**
2. True or False? The number of comparisons to insert $n$ elements into an AVL tree in the worst case is $\Theta(n \cdot \log\, n)$. **TRUE** only if the tree was initially empty.
3. Write a closed form expression for the running time of the function below as a function of $N$:

```
int foo (int N)
{
    if (N <= 0)
        return 1;
    else
        return 2 + foo(N/2);
}
```

   **ANSWER**: $\log_2 N + c$ for some constant $c$, or in order notation, $O(\log N)$

4. Write a pre-order traversal of a general tree whose nodes are defined by the following struct:

```
typedef struct _TreeNode {
    object element;
    TreeNode* leftChild;
    TreeNode* nextSibling;
} TreeNode;
void pre_order(TreeNode * t )
{
    if ( NULL != t ) {
        visit t->element;
        pre_order(t->leftChild);
        pre_order(t->nextSibling);
    }
}
```

5. Draw the binary search tree that results from inserting the following sequence of keys: $30, 40, 25, 60, 75, 23, 86, 12, 72$.

6. Using the same sequence, insert it into an AVL tree.

7. Delete 30 from the tree that you constructed in the above question.

8. What are the smallest and largest heights that an AVL tree with 50 nodes might have?

   **ANSWER: The largest height is 6. The smallest possible height is 5.**

9. Choose optimal values for B tree parameters, given a disk block size of 4KB, 32-bit addresses, key size of 16 bytes, and data size of 128 bytes.

   **ANSWER: m=205 and L=32**

10. Show the final state of a hash table of size 31 that uses the hash function $h(x) = x\%31$, with open addressing using linear probing, given that the following keys are inserted in order: $10, 20, 30, 40, 50, 60, 70, 80, 90, 100$.

    **ANSWER: There are no collisions. The table is**

| | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | 100 | 70 | 40 | 10 | | ... | 80 | 50 | 20 | | ... | | 90 | 60 | 30 |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | ... | 18 | 19 | 20 | 21 | ... | 27 | 28 | 29 | 30 |

11. Write a C++ function to implement a hash function using bit shifting using the middle square method if the word size is 8 bytes and the hash table is of size 4096.

```
hash( int x )
{
    return ( x * x ) >> 52;
}
```