CPSC 131 Midterm Exam Study Guide

# Fall 2020

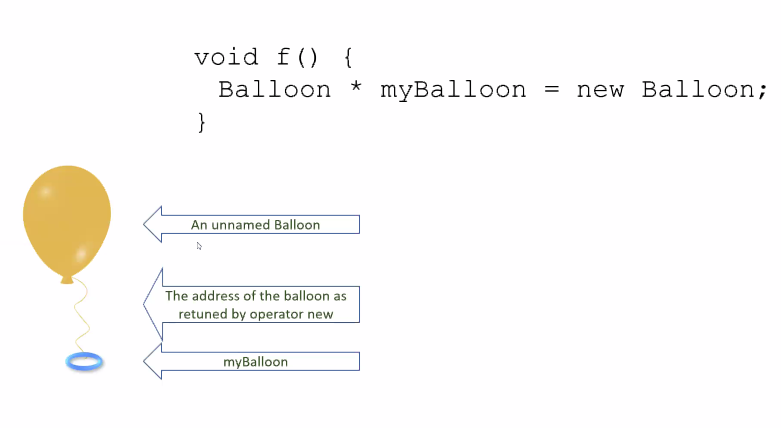
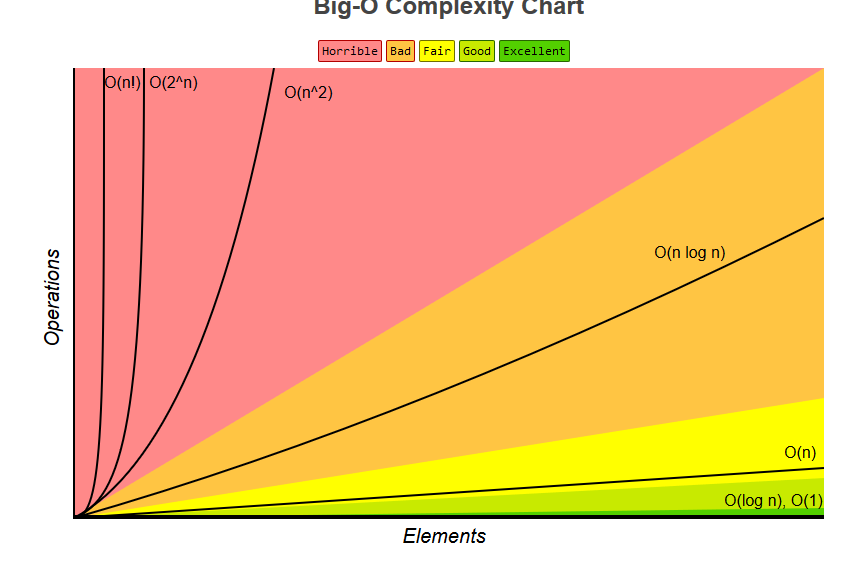
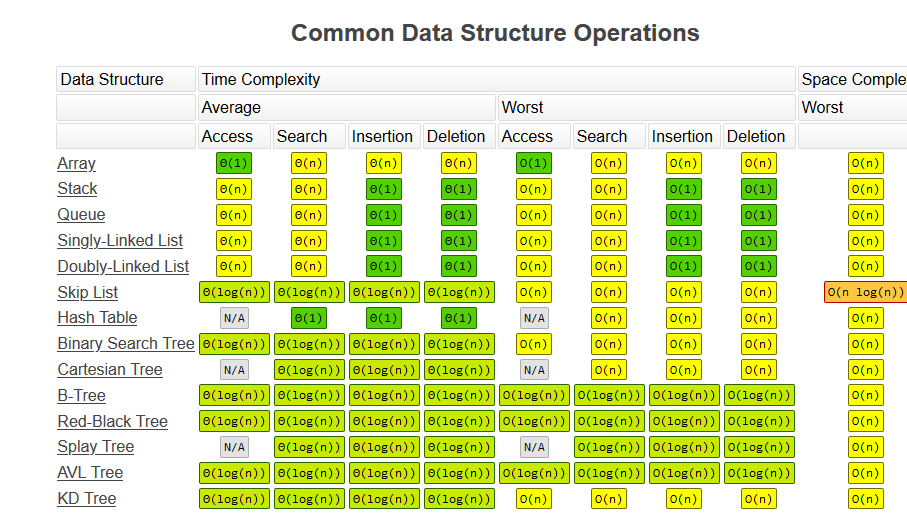
The midterm will take place 10/25.

You can expect the format of the exam to resemble that of our homework. Expect questions involving short answers, sketching data structures, stating big-O efficiency classes, and writing a short C++ function. The exam will be short enough that a prepared student should not feel any time pressure.

The exam is closed-books, closed-notes, no Internet access. The questions will be given to you on paper. You will add your answers to it.

The following material is fair game:

# Review of prerequisite material

1. P0:
   1. Arrays (for more info see [LEC 2](LEC2.docx))
      1. This is a container that is divided into elements that can store any data type
      2. It can be dynamic so that the number of rows and columns can change
      3. The init code is:  
         int arr[];
         1. In the case of pointers:  
            int\* arr[];
   2. pointers and references
      1. Here have a balloon  
         
      2. Pointer are variables that store the address.
         1. Init by:  
            int\* a;
         2. can be re-assigned any number of times
         3. can point nowhere (NULL)
         4. can be used for implementing data structures
      3. References are the addresses.  
         init by:  
         &a;
         1. always refers to an object
         2. be re-assigned after binding
         3. References can be used in function parameters and return types
      4. Both lead to the location on the memory of a variable.
      5. Be careful when you initialize them for hanging pointers lead to memory leaks.
      6. “Use references when you can, and pointers when you have to”
   3. dynamic memory
      1. when a dynamic object is created by a pointer, it is usually stored in a heap where it is freely available to be changed unlike most static forms of storage.
   4. C memory model (stack vs. heap)
      1. Stack
      2. Heap: free store
   5. Object-Oriented Programming (OOP)
      1. Goals, principles, and terminology
      2. writing classes with proper encapsulation (public/private)
      3. data members
         1. Constructors
         2. Destructors
         3. Setters
         4. Getters
         5. And other member functions
2. Recursion
3. Data structures and ADTs
4. fixed-length vector
5. singly linked list
6. doubly linked list
7. iterator ADT
8. Extendable vectors
9. Amortized Analysis
10. asymptotic analysis (big-O)
    1. 
11. deciding between O(1) and O(n)
12. efficiency classes for the fundamental operations of all the data structures we’ve covered (be able to work them out; don’t memorize them)
13. 
14. recursive list operations
15. stack ADT
16. stack based on a fixed-length vector
17. stack based on a singly linked list
18. queue ADT
19. queue based on a singly doubly linked list with dummy nodes
20. queue based on a fixed-length vector (a.k.a. circular array)