

CSCI 204 Exam 1 Study Guide

Fall 2017

1. The Rules

- You may bring one sheet of paper with anything you like hand-written on it, one or two sided. No printed information is allowed. Put your name on it.
- Any code or information on this sheet which comes from the Internet, a non-CSCI 204-course textbook, or another person needs a citation. Information from the Internet or from an outside book or person may only be used on this exam with citation and at the instructor's discretion. Always ask if you are unsure.
- Do not bring any electronic devices such as calculator, cell phone, tablet, or laptop. If you have to have one with you (such as a cell phone), turn the sound off and don't take it out of your bag.
- You hand in your helper sheet with your exam. I'll give it back later. I reserve the right not to grade your exam if you use but forget to include your helper sheet. If you don't use one, please indicate so on the cover sheet of the exam.
- The exam will begin at the hour and promptly end at the 52 minute mark. If I make an exam too long, I will deal with it later.
- Do not discuss the exam with anyone else except the instructor from the time it starts until after the exam is over. Do not discuss the exam with students not in our section before or after the exam until all exams are graded.

2. Exam Topics

Below is the list of major student learning outcomes for the first portion of the class. The exam will be based on this list.

A successful CSCI 204 student should be able to

- **In the area of basic Python,**
 1. Write basic Python programs as covered in CS203.
 2. Write and use functions for Python lists, strings, and dictionaries.
- **In the area of object oriented programming,**
 3. Explain how inheritance works and write code that uses it.
 4. Explain how class works and write a class that inherits another class.
 5. Explain the concept of overriding a parent method by writing one with the same name in the child class.
 6. Write code that calls a parent method from a child class.
 7. Write code that calls a parent method from a child class when the child has overridden that method.
 8. Use instance fields, a.k.a. member data or attributes, correctly.
 9. Call a parent constructor from the child constructor.
 10. Write a class constructor and the string function.
 11. Write and use properly common overloading operators such as '+', '>=', '<', '==', or 'in.'
 12. Explain exceptions.
 13. Use the try-except block and raise on existing exceptions.
 14. Write code that can open and read from file into a list or into a collection of lines.
- **In the area of ADTs, Arrays, singly linked list, doubly linked list,**

15. Explain what an ADT is.
 16. Write and use a linked node class.
 17. Write and use linked functions such as insert a node before or after another node, remove a node, traverse a linked list.
 18. Write your own functions for common tasks such as computing the length of a linked list, or the length of an array, or finding the location of a specific item in the array or in the linked list.
- **In the area of algorithm analysis,**
19. Explain the commonly used classes of complexity, $O(1)$, $O(\log n)$, $O(n)$, $O(n^k)$, $O(2^n)$ and any combinations of them.
 20. Perform Big-O analysis on Python code containing assignment, math, selection (`if-elif-else`), repetition (`for`, `while`), methods, and recursion.
 21. Perform Big-O analysis on Python list or string methods.
 22. Perform Big-O analysis on code containing arrays and linked nodes.

3. General format of the exam

The exam will have some short conceptual questions you'd answer or explain; some code comprehension where a segment of code is given and you explain the meaning and output of the code; and some Python functions or classes you'd write.

4. How to study for the exam

I suggest to form small groups and question each other the content of the course. In addition, I'd suggest to the following.

- Re-implement every ADT, every way. Then do a Big-O analysis on all those methods. Write program (the `main()` method) to test your ADTs.
- Write a class and another one that inherits it. Test your classes.
- Write some recursive methods. Do Big-O analysis on them.
- Make sure you are very comfortable with all the lab assignments. Make sure you understand the first programming project.