General Information

1. Prerequisites

CSE 100 has the following prerequisites: CSE 12, CSE 21 or Math 15B, and CSE 30 or other C or C++ experience. We expect everyone CSE 100 to know the following before taking the class:

- Designing and implementing code for basic data structures and ADTs such as linked lists, stacks, and queues.
- Performing running time analysis using Big-O notation
- Designing and implementing moderate-size programs in Java
- Testing and debugging programs in Java
- Writing C programs
- Debugging C programs using the gdb debugger
- Working with basic probabilities
- Writing proofs and basic understanding of asymptotic notations

2. Critical information, at a glance

You should read this entire syllabus. It is important. It may be the most important thing you read for this course. But here are the pieces of information you absolutely do not want to forget. I don't mean for this to sound scary, but so many students fail to read or understand these points, so I want to make them as clear as possible.

- Homework (PAs) is due by 11:59 pm on the due date. No late work will be accepted unless it is due to a documented emergency and the instructor of the course has to approve it before the due time.
- All reading assignments are due before the class for each session. We won't assign any points to grading.
- It is your responsibility to ensure that you have correctly submitted the correct code for your homework assignment. Incorrectly submitted assignments will be graded as is. We won't accept late work.
- All homework assignments must be done based on the instructions.
- All questions for the class should be posted to edstem. Emails to the instructor should be about personal and confidential matters only.

3. What will I learn in this class?

CSE 100 will teach you how to implement and analyze advanced data structures. It will also teach you skills for programming larger-scale programs in C++. Specifically, at the end of this course you will be able to:

- Select appropriate data structures to solve problems, considering their strengths and weaknesses
- Implement advanced data structures in C++, which might include any of: Binary Search Trees, Balanced Trees, KD Trees, Tries, various Hashing techniques, Heaps, Treaps, Graphs, and Priority Queues
- Critique different implementations of the same ADT/data structure
- Implement core algorithms that leverage advanced data structures, such as compression algorithms and graph algorithms
- Design and Debug medium-scale C++ programs
- Use the C++ STL effectively
- Use a version control system (git) for code management

- Derive best-case, worst-case and average-case running times of core data structures
- Compare empirical running times to theoretical expectations

4. Textbook: freely available from UCSD library.

Required Textbook:

Stepik: Introduction to Data Structures (Fall 2016) by Moshiri and Izhikevich (available at stepik.org)

Optional Textbook:

C++ For Java Programmers. Mark Weiss

Data Structures and Algorithm Analysis in C++, by Adam Drozdek, 4th edition