CMPT 125 - D100 Introduction to Computing Science and Programming II

Instructor(s): Igor Shinkar

Kalender Objective/Description:
A rigorous introduction to computing science and computer programming, suitable for students who already have some background in computing science and programming. Intended for students who will major in computing science or a related program. Topics include: fundamental algorithms; elements of empirical and theoretical algorithmics; abstract data types and elementary data structures; basic object-oriented programming and software design; computation and computability; specification and program correctness; and history of computing science.

Instructor's Objectives:
This course is a rigorous introduction to computing science, intended primarily for students who have already taken CMPT 120 as an introduction to algorithms and programming. Students will learn fundamental concepts of computing science and basic principles of algorithm design and software development. All case studies will be presented using a C-like subset of C++.

The co-requisite to this course is CMPT 127, which is a 3-credit lab course. Students who enroll in CMPT 125 will be manually enrolled in CMPT 127. Students are required to take both courses at once, but their grades will be assigned independently of each other.

CMPT 127 is a programming-intensive lab, where students are expected to complete approximately 100 assignments (C++ programs) over the course of 12 weeks. Assignments build in difficulty and duration from five minutes to two hours. Roughly 8-12 assignment problems will be posed per week: the first 1-3 of which will be completed with instructors and TAs in mandatory guided labs; the remainder will be completed as independent homework.

Prerequisites:
CMPT 120. Corequisite: CMPT 127. Students with credit for CMPT 126, 129, 135 or CMPT 200 or higher may not take for further credit. Quantitative

Topics:
- Brief review of elementary programming and problem solving; introduction to C.
- Performance measurements; algorithm design and analysis; asymptotics; fundamental algorithms.
- Recursion: simple recursion; recursion on trees; divide and conquer algorithms.
- Reasoning about programs: assertions, invariants, and correctness.
- Good coding style; defensive coding practices; testing.
- The memory model: addresses, dynamic data types, safe initialization, safe cleanup, and safe arrays.
- Encodings of basic types: int, unsigned, float, char, pointer.
- Compound data types; basic object/method design in C++.
- Abstract data types; information hiding; elementary data structures.
- Introduction to Social Issues

Grading:
To be discussed the first week of classes
Students must attain an overall passing grade on the weighted average of exams in the course in order to obtain a clear pass (C- or better).

**Academic Honesty Statement:**

Academic honesty plays a key role in our efforts to maintain a high standard of academic excellence and integrity. Students are advised that ALL acts of intellectual dishonesty will be handled in accordance with the SFU Academic Honesty and Student Conduct Policies (http://www.sfu.ca/policies/gazette/student.html).