

CS 1581: Honors Computer Science I (5)**Catalog Description:**

Similar to CS 1511: Computer Science I, but in greater depth and with more challenging assignments. For high-ability students.

Textbook: Hailperin, M., Kaiser, B., & Knight, K. (1999). *Concrete Abstractions: An Introduction to Computer Science Using Scheme*. Published on the web.

Course Goals:

This course is an introduction to computer science, focusing on the disciplines of procedural and data abstraction. It is intended for higher ability students, but it does not require experience with any particular kind of computer or programming language. It does require a minimum level of mathematical sophistication. This course is designed to be an introduction to the **science** of problem solving through the writing of computer programs. Students will become familiar with multiple computer programming paradigms, including functional, imperative, and object-oriented programming. Programming will initially be undertaken using the powerful, elegant, and high-level language Scheme. Scheme is simple to learn and helps students learn the principles of procedural and data abstraction that they will carry throughout their programming careers. The course concludes with an introduction to object-oriented programming using C++.

Prerequisites by Course & Topic:

3½ years of high school math and permission of the department.

Major Topics Covered in the Course:

- What is Computer Science?
- Recursion and Induction
- Iteration and Invariants
- Orders of Growth
- Higher-Order Procedures
- Compound Data and Data Abstraction
- Lists and Trees
- Generic Operations
- Stacks and Queues
- Procedural Programming in C++
- Object-Oriented Programming in C++

Class/Laboratory Schedule: Lecture: 3 hours per week, Discussion: 1, Laboratory: 1

Course Outcomes:

1. Understand the notion of a computational process and how to bring it about (*procedural* abstraction)
 - a. Describe computational processes with procedures written in high level languages (Scheme and C++).
 - b. Verify procedure correctness (reasoning by induction and testing).
 - c. Predict how long computational processes will take and how much memory they will require (big- Θ and big-O notation).
 - d. Increase the power of computational processes through higher-order procedures.
 - e. Exploit commonality to implement generic operations.
2. Understand how to describe and manipulate various types of data (*data* abstraction):
 - a. Construct compound data out of atomic data.
 - b. Create and use abstract data types (ADTs).
 - c. Separate ADT use from its implementation (information hiding).
 - d. Associate data with the operations performed on it (encapsulation).
3. Understand how to relate data to the state of the machine executing a computational process (abstractions of *state*):
 - a. Create and manipulate elementary data structures (lists, trees, stacks, and queues).

- b. Ensure the modularity of state by organizing computational entities into classes (object-oriented programming).

Relationship to Program Outcomes

This course is the first course in computer science. Students do not have to have prior programming experience. This course contributes to meeting the following program outcomes:

2. *Students can design, develop, and analyze significant software systems.*

This course introduces students to the basics of software design and analysis. Students learn a variety of data structures, the algorithms associated with them, and how to analyze algorithms. Course outcomes 1, 2, and 3 map to this program outcome.

3. *Students understand the fundamentals of computer organization and architecture, data structures and related algorithms, and programming languages.*

Students are introduced to the functional approach to procedure writing (Scheme), the procedural approach (C), and the object-oriented approach (C++). They are introduced to the efficiency of algorithms through big- Θ and big-O analysis. They manipulate and reason about lists, trees, stacks, and queues. Course outcomes 1-3 map to this program outcome.

4. *Students can apply computer science principles and practices to a variety of problems.*

Problem applications range from graphics (quilting designs), to security (digital signatures), to game playing (Nim), to databases (a movie query system). Course outcomes 1-3 map to this program outcome.

Assessment Plan for Course:

This course is assessed every other year by the instructor and a course assessment document covering all of the course outcomes and their effect on the program outcomes is prepared.

Estimate CSAB Category Content

	CORE	ADVANCED		CORE	ADVANCED
Data Structures	1		Computer Organization and Architecture		
Algorithms	1		Concept of Programming Languages	2	
Software Design	1				

Coordinator/Prepared by: T. Colburn