

# CS A100: SURVEY OF COMPUTER SCIENCE AND PROGRAMMING

Item	Value
Curriculum Committee Approval Date	10/16/2024
Top Code	070600 - Computer Science (Transfer)
Units	3 Total Units
Hours	90 Total Hours (Lecture Hours 36; Lab Hours 54)
Total Outside of Class Hours	0
Course Credit Status	Credit: Degree Applicable (D)
Material Fee	No
Basic Skills	Not Basic Skills (N)
Repeatable	No
Open Entry/Open Exit	No
Grading Policy	Standard Letter (S), • Pass/No Pass (B)

## Course Description

An introduction to the intellectual enterprises of computer science and the art of programming, for majors and non-majors, with or without prior programming experience. Topics include computational thinking, abstraction, algorithms, data structures, and computer science generally. This course teaches you how to program with an introduction to C, Python, SQL, HTML, CSS and JavaScript. Transfer Credit: CSU.

## Course Level Student Learning Outcome(s)

1. Students will be able to write programs in Scratch, C, Python, SQL, HTML, CSS and JavaScript.
2. Students will demonstrate understanding of the concepts of algorithmic analysis, artificial intelligence, secure coding and cybersecurity.

## Course Objectives

- 1. Demonstrate understanding of the CS concepts: inputs, outputs, data representation, abstraction, algorithms, running times and pseudocode.
- 2. Write programs in the Scratch programming language which employ functions, arguments, return values, variables, Boolean expressions, conditionals, loops, events and threads.
- 3. Use both the VS Code Graphical User Interface (GUI) and the Linux command line to write and compile programs using the C programming language.
- 4. Apply the following concepts to their C programs: header files, types, conditional, variables, loops, constants, comments, operators
- 5. Write simple cryptography programs using C arrays, strings and command-line arguments.
- 6. Write program which implement and use the following searching and sorting algorithms: linear and binary search, bubble, selection and merge sort.

- 7. Use Big O notation to perform asymptotic efficiency analysis of different algorithms.
- 8. Write programs which employ recursion as a problem-solving strategy.
- 9. Use pointers and dynamic memory allocation to write programs which process binary sound and image files.
- 10. Implement and apply the data structures: queue, stack, linked list, tree, binary-search tree, hash-table and tries.
- 11. Write programs using the Python programming language and 3rd-party libraries.
- 12. Describe concepts behind artificial intelligence: decision trees, Minimax, neural networks, large-language models and the transformer architecture.
- 13. Use the SQL language to create data models and query databases.
- 14. Understand how the Internet works, and write Web applications using HTML, CSS and JavaScript.
- 15. Use Python to create a complex Web application, making use of the Flask framework.
- 16. Understand and apply cybersecurity concepts of password managers, two-factor authentication and end-to-end encryption.

## Lecture Content

A. Introduction to CS 1001. Computer Science, input, output and binary numbers  
 2. Data representation, ASCII, Unicode, MIDI, RGB  
 3. Algorithms, linear and binary search  
 4. Pseudocode and problem solving  
 5. Scratch: abstraction, functions, conditional, loops and variables  
 B. Introducing the C Programming Language  
 1. Writing and understanding Hello World  
 2. Calling the built-in C functions  
 3. Creating variables to hold user information  
 4. Making decisions with if and else  
 5. Repeating actions with loops  
 6. Incorporating abstraction by writing your own functions  
 C. More on C Programming  
 1. Understanding the build process: Preprocessing, Compiling and linking  
 2. Using the VS Code debugger to understand your code  
 3. Creating and processing arrays to store collections of information  
 4. Using strings to store and process textual data  
 5. Writing programs which respond to command-line arguments  
 6. Cryptography and encryption  
 D. Algorithms and Data Structures  
 1. Searching for values: linear and binary search  
 2. Evaluating the running time of an algorithm  
 3. Creating user-defined types with struct  
 4. Sorting collections of data: bubble, selection and merge sort  
 5. Using recursion to solve problems  
 E. Using Memory  
 1. Pixels, hexadecimal and binary image representation  
 2. Strings, pointers, and pointer arithmetic  
 3. String comparison and string copying  
 4. Storing data on the heap with malloc  
 5. Using Valgrind to catch various memory errors  
 6. Using pointers with scanf and file I/O  
 F. Data Structures  
 1. Implementing and using stacks and queues  
 2. Using malloc and free to create dynamic structures  
 3. Implementing and using linked lists  
 4. Binary trees and binary search trees  
 5. Dictionaries, hash tables and tries  
 G. Introducing the Python Programming Language  
 1. Python, variables, conditionals. loops and types  
 2. Object-oriented programming and classes  
 3. Truncation and floating point imprecision  
 4. Python exceptions with raise, try and except  
 5. Searching and storing data with dictionaries  
 6. Command-line arguments and third-party libraries  
 H. Artificial Intelligence  
 1. An overview of different AI tools, vocabulary and concepts  
 2. Using CS50.ai or the rubber-duck debugger  
 3. Generative AI and decision trees  
 4. Tic-tac-toe and the minimax decision algorithm  
 5. Machine learning, deep learning and neural networks  
 6. Generative AI and large language models (LLM)  
 I. Storing Data  
 1. Flat-file databases, such as CSV  
 2. Relational databases and Structured Query Language

(SQL)3. Using SQL join to query data from different tables4. Optimizing queries by using indexes5. Using SQL in Python6. Race conditions and injection attacks J. Programming for the Internet1. How the Internet works: routers, DNS and HTTP2. Creating Web pages with Hypertext Markup Language (HTML)3. Using regular expressions to filter user input4. Using Cascading Style Sheets (CSS) to style your Web pages5. Using Bootstrap and other CSS frameworks6. Interactive Web pages with JavaScript K. Creating Interactive Web Applications1. Dynamic Web pages with the Flask Python framework2. Writing HTML forms to interact with the server3. Using Flask templates to control the page layout4. Handling GET and POST HTTP requests5. Using SQL with Flask6. Using sessions, Ajax and APIs L. Cybersecurity Topics1. Passwords and phone security2. Password managers and two-factor authentication3. Hashing, salt and rainbow tables4. Passkeys, encryption and file deletion

## Lab Content

Starting from Scratch Learning C Problems Hello, World. Hello, It's Me. Mario (less comfortable, more comfortable) Cash (less comfortable, more comfortable) Arrays and Strings Scrabble Readability Encryption (less comfortable, more comfortable) Algorithms Sort Plurality Elections (more less comfortable, very, very, very comfortable) Dynamic Memory Volume Filter (less comfortable, more comfortable) Recover Data Structures Inheritance Speller Python Hello Mario (less comfortable, more comfortable) Cash or Credit (less comfortable, more comfortable) Readability DNA SQL Songs Movies Fiftyville HTML, CSS and JavaScript Trivia Homepage Python Web Apps with Flask Birthdays Finance Final Project

## Method(s) of Instruction

- Lecture (02)
- DE Online Lecture (02X)
- Lab (04)
- DE Online Lab (04X)

## Instructional Techniques

Lecture, video shorts, demonstration, peer instruction

## Reading Assignments

Weekly lessons/videos for each topic. 1.5 hours per week.

## Writing Assignments

None

## Out-of-class Assignments

Weekly problem sets for each topic. 3 hours per week.

## Demonstration of Critical Thinking

Students will demonstrate their understanding of concepts such as algorithms, asymptotic analysis, cryptography and cybersecurity.

## Required Writing, Problem Solving, Skills Demonstration

Student will complete 10 sets of problem-solving, programming exercises during the semester, Each requiring 3-4 hours.

## Eligible Disciplines

Computer science: Master's degree in computer science or computer engineering OR bachelor's degree in either of the above AND master's degree in mathematics, cybernetics, business administration, accounting or engineering OR bachelor's degree in engineering AND master's degree in cybernetics, engineering mathematics, or business administration OR

bachelor's degree in mathematics AND master's degree in cybernetics, engineering mathematics, or business administration OR bachelor's degree in any of the above AND a master's degree in information science, computer information systems, or information systems OR the equivalent. Note: Courses in the use of computer programs for application to a particular discipline may be classified, for the minimum qualification purposes, under the discipline of the application. Master's degree required.

## Software Resources

1. Visual Studio Code. Microsoft/Github, Online Edition ed. Students will use the online IDE provided by Harvard University, in conjunction with Github and Microsoft, found at <https://cs50.div>

## Other Resources

1. Lecture materials will be presented via videos in Canvas