CS 2133: Computer Science II

Required Course: Required
Course Number: 2133
Course Name: Computer Science II
Credit Hours: 3
Lecture Hours: 3
Lab Hours: 1.5
Instructors: Dr. Christopher Crick

Book Title(s): Java: An Introduction to Problem Solving and Programming (8th Edition)
Book Author(s): Walter Savitch
Book Year(s): 2017

Course Description: Recursive algorithms. Intermediate methods of searching and sorting. Mathematical analysis of space and time complexity, worst case, and average case performance. Graphical user interfaces, event-driven programming, multithreaded program design, networking and the internet.

Course Prerequisites: CS 1113 (Computer Science I(A)) or equivalent

---

Course Goals: Students should be able to

- Design and implement object-oriented class hierarchies to model complex problems
- Build medium-sized event-driven Graphical User Interface (GUI) programs carefully designed according to Model-View-Controller architecture
- Interact using binary and text streams with filesystems and network sockets
- Understand and design applications around internet protocol specifications
- Implement simple multithreaded programs
- Understand the basics of algorithmic big-O analysis
- Choose appropriate data structures for particular tasks, based on algorithmic complexity

Student Outcomes: This class addresses the following student outcomes from the criteria for accrediting computing programs:

Student Outcome	Course Outcome
1	• Translate requirements documents into effective code-based solutions. • Refractor and reorganize code to increase clarity, maintainability and correctness.
2	• Use debugging strategies and tools effectively. • Implement test harnesses to evaluate program performance and adherence to requirements.
3	• Write code that communicates its purpose clearly and effectively, using comments and self-documenting variables, methods and organization. • Use, understand and produce API documentation for library classes.
4	• Understand how errors in software can have dangerous or fatal consequences, and use design strategies, tests, programming language features and tools to reduce the likelihood of such bugs surviving the coding processes
6	• Use appropriate design patterns such as Model-View-Controller to create complex GUI programs. • Identify and analyze resource constraints such as time, space and bandwidth. • Appropriately select data structures to minimize big-O resource use.

Course Topics:

Knowledge Area	Total Hours of Coverage
Algorithms and complexity (AL)	9
Architecture and Organization (AR)	1
Human-Computer Interaction (HCI)	4
Information Assurance and Security (IAS)	1
Networking and Communication (NC)	1.5
Operating Systems (OS)	1
Programming Languages (PL)	9.5
Software Development Fundamentals (SDF)	10
Software Engineering (SE)	2
Systems Fundamentals (SF)	0.5
Social Issues and Professional Practice (SP)	0.5

Knowledge Area	Knowledge Unit	Topics Covered	Hours
AL	Basic Analysis	All tier 1, analysis of iterative and recursive algorithms	2

AL	Algorithmic strategies	Divide-and-conquer, dynamic programming	2
AL	Fundamental Data Structures and Algorithms	All tier 1 and tier 2 except graphs and graph algorithms	3
AL	Basic Automata Computability and Complexity	Regular expressions, introduction to the P and NP classes and the P vs NP problem	1
AL	Advanced Data Structures Algorithms and Analysis	Balanced trees	1
AR	Digital Logic and Digital Systems	Overview and history of computer architecture, physical constraints	0.5
AR	Machine Level Representation of Data	Bits, bytes and words, signed and twos-compliment representations	0.5
HCI	Designing Interaction	Principles of graphical user interfaces, elements of visual design	1
HCI	Programming Interactive Systems	Software architecture patterns, event management and user interaction, widget classes and libraries, modern GUO libraries	3
IAS	Defensive Programming	Race conditions, correct handling of exceptions and unexpected behaviors	1
NC	Introduction	Roles of different layers	0.5
NC	Networked Applications	Naming and address schemes, HTTP as an application layer protocol, socket APIs	1
OS	Overview of Operating Systems	Role and purpose of operating systems	0.5

OS	Scheduling and Dispatch	Processes and threads	0.5
PL	Object-oriented Programming	All	4
PL	Functional Programming	First-class functions, defining higher-order operations on aggregates	0.5
PL	Event-Driven and Reactive Programming	All	3
PL	Runtime Systems	Dynamic memory management approaches and techniques	0.5
PL	Advanced programming Constructs	Lazy evaluation and infinite streams, control abstractions, string manipulation via pattern matching	1
PL	Concurrency and Parallelism	Constructs for thread-shared variables and shared-memory	0.5
SDF	Algorithms and Design	All	3
SDF	Fundamental Programming Concepts	All	3
SDF	Fundamental Data Structures	All	3
SDF	Development methods	Program correctness, simple refactoring, debugging strategies, documentation and programming style	1
SE	Software Design	System design principles, design paradigms, design patterns	1
SE	Verification and Validation	Testing fundamentals	1
SE	Computational Paradigms	Basic building blocks and components of computer, hardware as a computational paradigm	0.5

SE	History	History of computer hardware, software, networking, pioneers of computing, history of the Internet	0.5