



Appendix III - Program Outcomes using Indirect Assessment of Required Courses

CS 116	8	a	3	9	75%	Write, test, and debug simple recursive functions and procedures.
CS 116	9	ai	4	8	67%	Explain and apply object-oriented design and testing involving the following concepts: data abstraction, encapsulation, information hiding, inheritance, polymorphism
CS 116	10	ck	1	11	92%	Use a development environment to design, code, test, and debug simple programs, including multi-file source projects, in an object-oriented programming language.
CS 116	11	i	3	9	75%	Implement basic error handling
CS 116	12	a	1	11	92%	Solve problems by creating and using sequential search, binary search, and quadratic sorting algorithms (selection, insertion)
CS 116	13	j	3	9	75%	Determine the time complexity of simple algorithms.
CS 116	14	b	2	10	83%	Apply appropriate problem-solving strategies
CS 116	15	i	3	9	75%	Use APIs (Application Programmer Interfaces) and design/program APIs
CS 201	1	a	2	22	92%	Analyze and explain the behavior of simple programs involving the following fundamental programming constructs: assignment, I/O (including file I/O), selection, iteration, methods
CS 201	2	a	3	21	88%	Write a program that uses each of the following fundamental programming constructs: assignment, I/O (including file I/O), selection, iteration, methods
CS 201	3	ck	2	22	92%	Break a problem into logical pieces that can be solved (programmed) independently.
CS 201	4	j	3	21	88%	Develop, and analyze, algorithms for solving simple problems.
CS 201	5	i	2	22	92%	Use a suitable programming language, and development environment, to implement, test, and debug algorithms for solving simple problems.
CS 201	6	a	2	22	92%	Write programs that use each of the following data structures (and describe how they are represented in memory): strings, arrays
CS 201	7	a	3	21	88%	Explain the basics of the concept of recursion.
CS 201	8	a	4	20	83%	Write, test, and debug simple recursive functions and procedures.
CS 201	9	ai	4	20	83%	Explain and apply object-oriented design and testing involving the following concepts: data abstraction, encapsulation, information hiding, inheritance, polymorphism
CS 201	10	ck	4	20	83%	Use a development environment to design, code, test, and debug simple programs, including multi-file source projects, in an object-oriented programming language.
CS 201	11	i	2	22	92%	Implement basic error handling
CS 201	12	a	2	22	92%	Solve problems by creating and using sequential search, binary search, and quadratic sorting algorithms (selection, insertion)
CS 201	13	j	6	18	75%	Determine the time complexity of simple algorithms.
CS 201	14	b	3	21	88%	Apply appropriate problem-solving strategies
CS 201	15	i	4	20	83%	Use APIs (Application Programmer Interfaces) and design/program APIs
CS 330	1	a	0	27	100%	Illustrate by examples the basic terminology of functions, relations, and sets and demonstrate knowledge of their associated operations.
CS 330	2	a	0	27	100%	Demonstrate in practical applications the use of basic counting principles of permutations, combinations, inclusion/exclusion principle and the pigeonhole methodology.
CS 330	3	a	0	27	100%	Calculate probabilities of events and expectations of random variables for problems arising from games of chance.
CS 330	4	ab	5	22	81%	Establish and solve recurrence relations that arise in counting problems including the problem of determining the time complexity of recursively defined algorithms.
CS 330	5	a	2	25	93%	Model logic statements arising in algorithm correctness and real-life situations and manipulate them using the formal methods of propositional and predicate logic.
CS 330	6	a	6	21	78%	Outline basic proofs for theorems using the techniques of - direct proofs, proof by counterexample, proof by contraposition, proof by contradiction, mathematical induction.
CS 330	7	a	2	25	93%	Relate the ideas of mathematical induction to recursion and recursively defined structures.
CS 330	8	j	0	27	100%	Illustrate by example basic terminology of graph theory and model problems in computer science using graphs and trees.
CS 330	9	j	1	26	96%	Deduce properties that establish particular graphs as Trees, Planar, Eulerian, and Hamiltonian.
CS 330	10	j	0	27	100%	Illustrate the application of trees and graphs to data structures.
CS 330	11	j	3	24	89%	Explain the basic concepts modeling computation including formal machines, languages, finite automata, Turing machines
CS 331	1	bj	2	42	95%	Explain, implement, and apply the following data-structures: \n- lists (unordered and ordered), stacks, queues, expression trees, binary search trees, heaps, and hash tables.
CS 331	2	cj	5	39	89%	Analyze the time and space complexity of algorithms using asymptotic upper bounds (big-O notation).
CS 331	3	bj	0	44	100%	Explain and use references and linked structures.
CS 331	4	k	6	38	86%	Outline basic object-oriented design concepts: composition, inheritance, polymorphism.
CS 331	5	cj	6	38	86%	Write and test recursive procedures, and explain the run-time stack concept.
CS 331	6	ij	2	42	95%	Analyze searching and sorting algorithms, and explain their relationship to data-structures.
CS 331	7	ij	9	35	80%	Choose and implement appropriate data-structures to solve an application problem.
CS 331	8	dik	9	35	80%	Explain how to use unit tests and version control in your software development.

Appendix III - Program Outcomes using Indirect Assessment of Required Courses

CS 350	1	a	2	8	80%	The focus of this course will be how computers work with particular focus on the relationship between software written in a high level language and the computer systems that compile and execute them. Specifically, we will be uncovering details of the lower layers of abstraction present in computer systems by discussing topics which include:
CS 350	2	c	2	8	80%	Digital Logic Hardware
CS 350	3	a	2	8	80%	Low Level Data Representation
CS 350	4	c	0	10	100%	von Neumann Computing Model
CS 350	5	c	0	10	100%	Instruction Set Architectures
CS 350	6	c	2	8	80%	Assembly Language
CS 350	7	c	2	8	80%	Input/Output
CS 350	8	c	3	7	70%	Traps and Subroutines
CS 350	9	c	3	7	70%	Interrupts
CS 350	10	c	4	6	60%	Call Stack
CS 350	11	c	2	8	80%	C programming
CS 350	12	ij	0	10	100%	We will study these concepts in the context of the LC-3 processor as well as the ARM architecture.
CS 350	13	bdf	1	9	90%	We will explore the relationship between hardware and software by programming an ARM-based device, the Nintendo Gameboy Advance in C. Students will gain an understanding of all the components of a computer, insight into the interactions between software and hardware, and an appreciation for the advantages and limitations of the abstractions provided by higher level languages.
CS 351	1	a	3	27	90%	Define the concept and role of a process in a modern operating system
CS 351	2	a	3	27	90%	Describe the key abstractions an operating system provides to running processes
CS 351	3	cj	5	25	83%	Describe the function, usage, and operation of system calls related to process management, memory management and I/O
CS 351	4	cj	2	28	93%	Explain exceptional control flow, including:\n- Hardware interrupts\n- Software exceptions / Traps\n- Signals and signal handling
CS 351	5	cj	2	28	93%	Describe the essential operation of a modern MMU from a programmers standpoint, including:\n- Caching and the TLB\n- Segmentation and paging for virtual memory
CS 351	6	cj	5	25	83%	Explain the operation of various memory allocation methods, including:\n- Implicit allocation (garbage collection)\n- Explicit allocation (malloc/free, reference counting, etc.)
CS 351	7	bi	8	22	73%	Describe, utilize, and implement a dynamic memory allocation API.
CS 351	8	bi	9	21	70%	Describe and utilize the system-level I/O API of a modern operating system, including:\n- File descriptors\n- File I/O\n- Buffered I/O\n- Interprocess communication
CS 351	9	bi	11	19	63%	Describe and utilize a low-level socket based networking API. This should include:\n- Client / Server model\n- Internetworking\n- Berkeley sockets
CS 351	10	bi	13	17	57%	Describe, design and utilize concurrent programming APIs, including:\n- POSIX Threads\n- Re-Entrant code\n- Synchronization primitives
CS 430	1	a	2	27	93%	Use big O, omega, and theta notation to give asymptotic upper, lower, and tight bounds on time and space complexity of algorithms.
CS 430	2	ab	1	28	97%	Determine the time complexity of simple algorithms, deduce the recurrence relations that describe the time complexity of recursively defined algorithms, and solve simple recurrence relations.
CS 430	3	cj	3	26	90%	Design algorithms using the brute-force, greedy, dynamic programming, divide-and-conquer, branch and bound strategies.
CS 430	4	cj	2	27	93%	Design algorithms using at least one other algorithmic strategy from the list of topics for this unit.
CS 430	5	cj	4	25	86%	Use and implement the fundamental abstract data types -- specifically including hash tables, binary search trees, and graphs -- necessary to solve algorithmic problems efficiently.
CS 430	6	i	1	28	97%	Solve problems using techniques learned in the design of sequential search, binary search, O(N log N) sorting algorithms, and fundamental graph algorithms, including depth-first and breadth-first search, single-source and all-pairs shortest paths, and at least one minimum spanning tree algorithm.
CS 430	7	bc	3	26	90%	Demonstrate the following abilities: to evaluate algorithms, to select from a range of possible options, to provide justification for that selection, and to implement the algorithm in simple programming contexts.
CS 430	8	fh	2	27	93%	Communicate theoretical and experimental analyses of a set of algorithms (i.e. sorting) in a lab report format.
CS 430		l	4	18	82%	UG B or better grades
CS 440	1	aj	1	12	92%	Explain major classes of programming languages: techniques, features, and styles. \n- Know how to use boxed and unboxed variables \n- Be able to use higher order functions.
CS 440	2	aj	2	11	85%	How to specify formally the meaning of a language --- to people and to the computer. \n- Use Transition, Typing, and Denotational Semantics to define a language construct. \n- Be able to specify the language of regular expressions. \n- Determine if a grammar is LL, and write a parser for it using recursive descent. \n- Determine if a grammar is LR, and write a parser for it using a parser generator. \n- Describe the algorithm for both LL and LR parser generation.
CS 440	3	ci	0	13	100%	Explain Recursion \n- Know how to use both tail recursion and standard recursion. \n- Know how to use higher order functions to eliminate recursion.

Appendix III - Program Outcomes using Indirect Assessment of Required Courses

CS 440	4	ci	1	12	92%	Explain Abstraction \n- Know how to create user-defined types. \n- Know how to use functions to model integers. \n- Know how to use trees to model language constructs.
CS 440	5	ci	0	13	100%	Explain Transformation \n- Know how to interpret a language. \n- Know how to use unification.
CS 440	6	ci	1	12	92%	How to choose a language.
CS 440	7	ci	4	9	69%	How to implement a language.
CS 440	8	h	0	13	100%	Emphasis: learn theory and apply it.
CS 440		l	3	13	81%	UG B or better grades
CS 450	1	aj	0	28	100%	Explain the range of requirements that a modern operating system has to address.
CS 450	2	aj	0	28	100%	Define the functionality that a modern operating system must deliver to meet a particular need.
CS 450	3	aj	3	25	89%	Articulate design tradeoffs inherent in operating system design.
CS 450	4	aj	5	23	82%	Explain the concept of a logical layer.
CS 450	5	aj	6	22	79%	From the perspective of building operating systems, explain the benefits of building these layers in a hierarchical fashion.
CS 450	6	aj	1	27	96%	Describe how the resources of the computer system are managed by software.
CS 450	7	aj	6	22	79%	Relate system state to user protection.
CS 450	8	aj	0	28	100%	Justify the presence of concurrency within the framework of an operating system.
CS 450	9	aj	0	28	100%	Demonstrate the potential run-time problems arising from the concurrent operation of many (possibly a dynamic number of) tasks.
CS 450	10	aj	3	25	89%	Summarize the range of mechanisms (at an operating system level) that can be employed to realize concurrent systems and be able to describe the benefits of each.
CS 450	11	aj	1	27	96%	Explain the different states that a task may pass through and the data structures needed to support the management of many tasks.
CS 450	12	aj	0	28	100%	Compare and contrast the common algorithms used for both preemptive and non-preemptive scheduling of tasks in operating systems.
CS 450	13	aj	5	23	82%	Describe relationships between scheduling algorithms and application domains.
CS 450	14	aj	1	27	96%	Investigate the wider applicability of scheduling in such contexts as disk I/O, networking scheduling, and project scheduling.
CS 450	15	aj	7	21	75%	Introduce memory hierarchy and cost-performance tradeoffs.
CS 450	16	aj	3	25	89%	Explain what virtual memory is and how it is realized in hardware and software.
CS 450	17	aj	7	21	75%	Examine the wider applicability and relevance of the concepts of virtual entity and of caching.
CS 450	18	aj	8	20	71%	Evaluate the trade-offs in terms of memory size (main memory, cache memory, auxiliary memory) and processor speed.
CS 450	19	aj	3	25	89%	Defend the different ways of allocating memory to tasks on the basis of the relative merits of each.
CS 450	20	aj	7	21	75%	Summarize the features of an operating system used to provide protection and security, and describe the limitations of each of these.
CS 450	21	aj	1	27	96%	Summarize the full range of considerations that support file systems.
CS 450		l	8	20	71%	UG B or better grades
CS 485	1	eg	0	9	100%	Demonstrate an understanding of the social and professional context in which computing is done.
CS 485	2	eg	0	9	100%	Demonstrate an understanding of the basic cultural, social, legal, and ethical issues inherent in the discipline of computing.
CS 485	3	g	1	8	89%	Identify milestones in the development and application of information technology.
CS 485	4	eg	0	9	100%	Ask serious questions about the social impact of computing and to evaluate proposed answers to those questions.
CS 485	5	e	1	8	89%	Demonstrate an awareness of the basic legal rights of software and hardware vendors and users, and the ethical values that are the basis for those rights.
CS 485	6	e	0	9	100%	Research the social and ethical issues of a computer related topic from the list in the syllabus.
CS 485	7	df	0	9	100%	Communicate, both orally and in written form, social and ethical issues of a computer related topic from the list in the syllabus.
CS 485		l	3	19	86%	UG B or better grades
CS 487	1	k	3	41	93%	Understand and explain software development as a series of engineering activities, and processes.
CS 487	2	dk	3	41	93%	Demonstrate software development team-working skills.
CS 487	3	bc	6	38	86%	Analyze client/user needs.
CS 487	4	bc	5	39	89%	Select an appropriate life cycle and process model for development of a software product.
CS 487	5	bc	5	39	89%	Explain the importance of software quality evaluation activities.
CS 487	6	bc	3	41	93%	Develop a series of software life-cycle deliverables.
CS 487	7	bc	3	41	93%	Develop representations/models and descriptions of an evolving software product for inclusion in a requirements specification document.
CS 487	8	bc	6	38	86%	Build a multi-level design model and evaluate software design alternatives
CS 487	9	bc	7	37	84%	Design, execute, and log multi-level software tests.
CS 487	10	i	7	37	84%	Describe the role that tools can play in the software life cycle.
CS 487	11	defh	2	42	95%	Communicate, verbally and in writing, the deliverables of a software development project.
CS 487		l	3	24	89%	UG B or better grades