Math 273 Course Content and Objectives

COURSE CONTENT AND SCOPE - **Lecture:** Outline the topics included in the lecture portion of the course *(Outline reflects course description, all topics covered in class).*	Hours Per Topic	COURSE OBJECTIVES - **Lecture:** Upon successful completion of this course, the student will be able to...*(Use action verbs - see Bloom's Taxonomy for 'action verbs requiring cognitive outcomes.')*
Review of Object-oriented programming.	4	Understand abstraction and classes, class constructors and deconstructors, arrays of objects, inheritance, dynamic allocation, polymorphism, abstract base classes and interfaces, and design issues. Write a series of classes that utilize polymorphism.
Further topics in programming and Object-oriented programming.	3	Apply multiple inheritance, virtual inheritance, the diamond problem, macros, class and function templates, and multithreading. Create a class template.
Event-driven programming.	5	Create an effective graphical user interface, capture and process messages, and use application programming interfaces. Create an event-driven program that handles all appropriate messages.
Video game programming.	5	Create graphics loops, video graphics card capabilities, multiple buffering, sprites, animation, and capturing user input. Choose the graphics application programming interface from DirectX, OpenGL, or any other professionally used application programming interface. Write a program that demonstrates animation.
Complexity analysis.	3	Analyze computational complexity, big-O notation, best case analysis, worst case analysis, average case analysis, amortized analysis, and NP-completeness. Analyze the best, worst, and average case complexities of an algorithm.
Linked lists.	5	Implement singly linked lists, doubly linked lists, circular lists, skip lists, sparse tables, and linked lists in the Standard Template Library. Implement a circular list.
Stacks and queues.	4	Implement stacks, queues, and priority queues. Run simulations using queues. Create stacks and queues in the Standard Template Library.
Recursion.	5	Understand recursion in mathematics and how computers perform function calls. Apply mathematical recursion, tail recursion, nontail recursion, indirect

		recursion, and nested recursion appropriately. Understand limitations and appropriate uses of recursion and backtracking. Determine situations in which recursive algorithms are appropriate and be able to write recursive implementations for those situations.
Binary trees.	5	Use trees, binary search trees, searching binary trees, and tree traversal. Create algorithms for insertion, deletion, and balancing. Use heaps and implement priority queues, and expression trees. Analyze the complexity of algorithms associated with the tree.
Sorting.	5	Implement insertion sort, selection sort, bubble sort, shell sort, heap sort, Quicksort, merge sort, radix sort, and sorting in the Standard Template Library. Analyze and compare complexity of sorting algorithms.
Hashing.	5	Implement hash functions, division, folding, mid-square function, extraction, radix transformation, collision resolution, and a Map class template.
Graphs.	3	Represent graphs, graph traversals, shortest paths, cycle detection, spanning trees, and connectivity. Write a program that detects cycles in a graph.
Final examination.	2	Final examination.
Total:	54	
Total Lecture Hours In Section I Class Hours:	54	

Lab

		Bloom's Taxonomy
Review of objects and classes.	2	Develop and create a Scanner class that parses text for specified tokens.
Further topics in programming and Object-Oriented Programming.	2	Develop and create a class template that models complex numbers with integer, fractional, or decimal coefficients.
Event-driven programming.	2	Develop and create an event-driven program with a graphical user interface that computes U.S. federal income tax.
Video game programming.	2	Develop and create a program that animates a character that moves according to user input.
Complexity analysis.	2	Develop and create a program implements the binary search algorithm.
Linked lists.	2	Develop and create a program that models the behavior of opposing traffic signals.
Stacks and queues.	2	Simulate customer wait time at a restaurant using queues.
Recursion.	2	Develop and create a program that parses a grammar using the method of recursive descent.
Binary trees.	2	Develop and create an interpreter that parses and evaluates mathematical expressions that may include previously defined variables.
Sorting.	2	Develop and create programs that perform polynomial arithmetic.
Hashing.	2	Implement a Map class template using hashing.
Final project.	14	Design, develop, and write a large scale program in a collaborative environment. The program may use ideas from event-driven programming or video game programming, but must use at least one data structure, searching, or sorting algorithm discussed in the course in a significant way.
	36	
	36	