Math 273 Course Content and Objectives

Lecture: M gl c cdca am c bc a gn gml mmga am c cb gl a ',	Hours Per Topic	Lecture: c a gml c cc mm ml m dm a gml c co ggle amel gg c m am c ,'
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.		

Lab

Lab: Hours

M gjc cdca am c bc a gn gml mmga
amc cbgj a ',