

CIS 121 Introduction to Programming Languages II

Course Number & Title (A.1)	CIS 121 Introduction to Programming Languages II
Credit Units (A.2)	1 CU (3 hours of lecture per week)
Instructor (A.3)	Kostas Daniilidis, Professor, kostas@cis.upenn.edu , 472 Levine Hall, 215-898-8549
Text(s)/Required Materials (A.4)	Only recommended: Mark A. Weiss, Data Structures and Problem Solving Using Java, 4/E, Addison-Wesley, 2010 Course notes made available via web page.
Catalog Description (A.5a)	This is a course about Algorithms and Data Structures using the JAVA programming language. We introduce the basic concepts about complexity of an algorithm and methods on how to compute the running time of algorithms. Then, we describe data structures like stacks, queues, maps, trees, and graphs, and we construct efficient algorithms based on these representations. The course builds upon existing implementations of basic data structures in JAVA and extends them for the structures like trees, studying the performance of operations on such structures, and their efficiency when used in real-world applications. A large project introducing students to the challenges of software engineering concludes the course.
Prerequisites (A.5b)	CIS 160
Course Satisfies (A.5c)	[] Math [] Science [x] Engineering [] Technical Elective [] TBS (check only one, UG curric impact only) Required
Course Web	http://www.seas.upenn.edu/~cis121
Course Outcomes (A.6a)	<ul style="list-style-type: none"> • Ability to make proofs using just definitions of Big-Oh, -Omega, -Theta • Ability to find the running time of a given code segment • Given a simple problem find an algorithm to solve it in prespecified Big-Theta • Ability to understand geometric progressions and apply it to heaps and trees • Ability to discover a recurrence relation that will simplify computations of number of nodes and leaves in balanced trees • Ability to use mathematical properties of graphs to analyze algorithms • Ability to run and understand search algorithms • Modelling of capital gain computation using queues • Use of quadtrees in achieving image compression • Huffman coding for optimal text compression • Using only JAVA interface of data-structures with hidden implementation • Understand how to embed results in a dynamic web page using javascript and the google maps API • The power of a data structure as a mathematical representation (graphs in finding degree of separation) • The final project involved application of shortest paths and minimum spanning trees using data from an open source (openstreetmap.org) where students had to deal with all data import issues as well as make a good use of memory.
Contribution towards Program Outcomes (A.6b)	a,b,c,i,j,k
Topics Covered (A.7)	<ul style="list-style-type: none"> • <input type="checkbox"/> Asymptotic complexity, big-oh notation, code and pseudocode analysis. • <input type="checkbox"/> Stacks, queues, lists. • <input type="checkbox"/> Applications of stacks to expression processing. • <input type="checkbox"/> Priority queues, adapter design pattern, heaps. • <input type="checkbox"/> Tree traversals, iterator design pattern. • <input type="checkbox"/> Search trees, balanced trees. • <input type="checkbox"/> Hash tables. • <input type="checkbox"/> Graphs, depth-first and breadth-first search. • <input type="checkbox"/> Weighted graphs, shortest paths (Dijkstra's algorithm), minimum spanning trees (Kruskal's algorithm) • <input type="checkbox"/> Digraphs, DAGs, topological sorting, transitive closure • <input type="checkbox"/> Elements of software engineering: process models, XP, "agility", UML, refactoring.
Grading Details	homeworks+quiz (30%), the two midterms (2 X 15%), the final (30%), and lab participation (10%)
Prepared By/Date	Kostas Daniilidis, January 2011