COURSE SYLLABUS

COP 4530, Sections 1 and 2
Data Structures, Algorithms, and Generic Programming

Fall Semester 2006

Prerequisites:

COP 3330: Object-Oriented Programming, and *MAD 2104: Discrete Mathematics*. Pre or Corequisite: *CDA 3100: Computer Organization I*. The pre-requisites will not be waived.

Class Schedule:

Activity	Day	Time	Location
Lecture - Sec 1-2	TR	9:30 am - 10:45 am	MCH 301
Lecture - Sec 3-4	TR	12:30 pm - 1:45 pm	LOV 301
Recitation - Sec 2	M	9:05 am - 9:55 am	MCH 128
Recitation - Sec 3	M	10:10 am - 11:00 am	MCH 128
Recitation - Sec 4	M	11:15 am - 12:05 pm	MCH 128

Contact information:

Instructor: Ashok Srinivasan

Office hours: TW 2 - 3 pm. I am also usually available in my office, and you can feel free to meet me in the afternoons, except before class Tuesday. Alternatively, you may schedule an appointment, either by email or by phone.

Office: 169, Love Building

Phone: 644-0559

Email: asriniva@cs.fsu.edu

Teaching Assistant:	Huey Ling Toh	Jane Ren
Office hours:	M 12:30 pm - 1:30 pm	T 3:20 pm - 4:20 pm
Office:	CS Majors lab	422 Dirac Science Library
Phone:	TBA	644-8531
Email:	toh@cs.fsu.edu	jingyren@cs.fsu.edu

Course material:

Required Material:

- Adam Drozdek, *Data Structures and Algorithms in C++*, third edition, Thomson Course Technology 2005, ISBN 0 534-49182-0.
- Online lectures material: available through Blackboard (http://campus.fsu.edu).
- Source code for example programs from the text is available at www.mathcs.duq.edu/drozdek/DSinCpp
- Other code: Additional code will be made available under the /home/courses/cop4530/fall05 directory and subdirectories on the Computer Science file server.

Optional reference material:

- Deitel, H.M. and Deitel, P.J. (1998). *C++ How to Program (2nd ed.)*. New Jersey: Prentice Hall. ISBN 0-13-528910-6.

  This is the text book used in COP 3330. It will help you review C++ features that you learned in COP 3330, which you will need in this course too.

Online resources:

- Emacs Reference Card: http://www.indiana.edu/~ucspubs/b131
- Emacs tutorial: www.math.utah.edu/computing/unix/emacs.html
- DJGPP - DOS GNU Unix: http://www.delorie.com/djgpp
- FSU User Services Site Licenses: http://sl.us.fsu.edu/index.html

Computer accounts:

- You will need a computer account in the Computer Science department. Please follow the procedure outlined in http://www.cs.fsu.edu/sysinfo/newstudent.html to obtain an account, if you do not have one already.
- Class email will be sent to your ACNS account (@garnet.fsu.edu). So please obtain an ACNS account, if you do not have one already. If you use another email account (such as yahoo), then you **must** forward your garnet email to that account. Instructions on obtaining an account and forwarding email are available at https://cars.acns.fsu.edu.
- You will need to use Blackboard (http://campus.fsu.edu). You will be able to access it only if you have an ACNS account.

Course rationale:

So far, you have acquired proficiency in programming. This course will start the process of your transformation from a programmer to a computer scientist. One of the important tasks of a computer scientist is to make efficient use of computational resources. This course will teach you about different ways of organizing the data to facilitate such

efficient use, and will also discuss efficient techniques to perform some fundamental operations in computer science. A subsequent course on Algorithms, COP 4531, will teach you to use the techniques discussed in this course to solve commonly encountered computer science problems efficiently. Both these courses will also teach you to analyze the efficiency, and prove the correctness, of your program in a mathematically rigorous manner. Material you learn in these two courses is critical to your becoming a good software developer later.

Course description:

This IS NOT a course in object-oriented programming!

This is a course about efficiency of programs and programming. In this course, we will pursue various meanings of "efficient". It is efficient to:

- minimize the *run time* of code, especially code that is destined for re-use.
- minimize the memory and storage needs of code, recognizing that there may be a trade-off between speed and memory requirements.
- spend less time writing a program of equal quality. In fact, it is even more efficient to spend the same time writing a program of higher quality.
- re-use code, instead of re-writing code.
- select only the linguistic features you need without having to use costly extra features you do not need.

Furthermore, in many applications, correctness is the ultimate form of efficiency, while in others efficiency means getting the best result possible in the limited time (or space) available.

Efficiency can happen at different levels. Take code:

- Source code can be small in size, easy to read, and easy to understand.
- Executable code can be fast or compact (or both).
- The code production process can be efficient by applying good software engineering methodology; savings in human effort too represent efficiency. Effort can be saved by good design, by careful (error-free) programming, and by re-using both code itself and patterns of problem solving that are known to be successful.
- Code can run efficiently, in either a temporal or spatial sense.

All these ideas of efficiency are central to this course. It is also true that all of these ideas of efficiency are fundamental to the design and specification of the C++ language, which is one of many reasons C++ is a good choice for the core language in our curriculum and for this course.

The three topics mentioned in the title of the course are:

Data structures

We will discuss data structures in abstract terms, as *abstract data types* (ADTs), but we will also implement them concretely, using C++.

Algorithms

Algorithms are formalizations of processes that result in predictable and desirable outcomes. They are used in a variety of contexts. Particularly, data structures are made usable by implementing algorithms for searching, sorting, and indexing the structures.

Generic programming

Generic programming is the science of component re-use. We will explore coding for re-use of both data structures and algorithms in C++. *Coding for re-use* and *re-use of code* are important aspects of software engineering.

We will also have several substantial programming projects that involve the implementation and use of data structures, algorithms, and generic programming.

Learning objectives:

At the end of this course, you should be able to accomplish the objectives given below. Furthermore, since the tasks mentioned below are those that every computer scientist is expected to be familiar with, it will help you if you learn it well enough that you have a permanent working knowledge of the material discussed.

Data Structures

- Define and use the following *abstract data types* (ADTs) as generic containers:
  Positional ADTs: vector, list, deque, stack, queue, graph, digraph
  Associative ADTs: table, map (associative array), priority queue, set
- Implement these ADTs, including performance constraints (in terms of runtime complexity) on the operations.

  Note that this implies the detailed study of trees of several types as implementation structures and the use of template classes as well as the elementary study of algorithms and their complexity.

Algorithms

You should be able to accomplish the following:

- Show the steps performed by algorithms that use the data structures given above, and of their simple variants.
- Prove the correctness of algorithms that use the data structures given above, and of their simple variants.
- Perform time and space *complexity analysis* of algorithms that use the data structures given above, and of their simple variants.

Generic Programming

You should be able to accomplish the following:

- Implement a given data structure as a *generic container*, using class templates with typename template parameters.
- Implement a given algorithm generically, using function templates with iterator template parameters.

Attaining these objectives will enable you to:

- Write code that performs many of the fundamental operations of Computer Science efficiently.
- Modify the techniques we discuss, when an application on which you are working is unable to use the techniques directly.
- Write code that is re-usable.

Your responsibilities:

Deadlines and instructions

Following the same professional guidelines that you will encounter in business, there are strict deadlines, and instructions that must be followed. Please read instructions carefully, and schedule your activities so that you submit assignments well in time. You should check your **garnet** email account and the class web page regularly, and note other announcements, on-line and in class.

Class participation and quizzes

I will ask you questions in class: (i) review questions on the previous lecture, and (ii) questions on the material currently being discussed, in order for me to obtain feedback on how well you understand the material. You should be prepared to answer these questions, and should also participate by asking questions, suggesting ideas, and performing in-class assignments that I give. Of course, you cannot participate in class unless you attend it! The recitations too form an important component of the course. You will be given quizzes during most recitations, on material covered since the previous quiz, and you will also be asked to write small programs and perform other tasks. You will be graded on these too.

Group work

There will be two types of group work in this course. (i) *Study groups:* We will organize study groups for those who are interested in this. Interaction with your group can assist you in understanding lecture material, and in preparing for lectures. More details on this are available at: www.cs.fsu.edu/~asriniva/courses/DS05/studygroups.html. (ii) *Group assignments:* You will have at least one group programming assignment. You should participate in your group's activities and make a fair contribution to the effort of your group. You can learn more about working in groups at: www.cs.fsu.edu/~asriniva/courses/DS05/groupwork.html.

Reading assignments

After each lecture, you will be given a reading assignment from the text and from the online lecture notes, pertaining to that lecture. You should read these, and also practice with example codes that we supply. New material builds on the old ones. So, if you have trouble with some material, please get help through the discussion board on Blackboard, or from a teaching assistant or me, before the next class. You should also peruse the material for the next lecture, and be prepared to answer questions on it, which I will provide in advance. I expect that you will need to spend between one and two hours studying, for each lecture. The programming assignments and exams will consume *additional* time. The following learning components are important, and you may want to verify if you do satisfactorily on these, after studying the material.

- *Knowledge*: Do you understand the terminology used? Given a data-structure and some data or operations, can you tell how the data is represented? Given an algorithm and its input, can you describe the steps carried out by the algorithm and the output? Given a data structure or algorithm, can you write C++ code for it? Given a data-structure and some operations on it, or an algorithm, can you give the time and space complexity?
- *Understanding*: If some aspect of a data-structure, an operation on it, or an algorithm, were modified, can you analyze the time complexity? If some aspect of an operation on a data-structure, or an algorithm, were modified, can you prove or disprove its correctness? In order to answer these questions, you need to understand how each component of an algorithm affects the time complexity, and why each component of an algorithm is important for its correctness. After you learn about what an algorithm does (and have, thus, acquired "knowledge"), it will be useful for you to think of different things that can be changed, and see how that will affect the time complexity or correctness.
- *Application*: Given a real life, or artificial, problem, can you decide on suitable data structures and algorithms from those we have studied, to solve that problem?
- *Creativity*: Can you modify algorithms that we have studied, to make them more efficient for special situations? Given a problem for which our algorithm is not valid as designed, can you modify the algorithm to solve the problem, and then prove the correctness of your solution, and analyze its time complexity? Can you use multiple data structure and algorithms from those that we have studied, to solve a new problem?

Homework assignments

You will have four programming assignments and two written assignments in this course, and you will have around ten days to work on each one. At least one of these (the third programming assignment), will be a group project. The projects will be announced on the Blackboard course web site under "Assignments". We will also discuss it during the recitations. Assignment submission instructions are

available at www.cs.fsu.edu/~asriniva/courses/DS05/HWinstructions.html. The programming assignments will be **substantially** more difficult than those in previous programming courses, and require substantially more time and effort. Please start working on the assignments as soon as they are announced, if you wish to complete them!

Course calender:

Week	Lecture	Chapter	Assignments
1	Recitation	Syllabus, initial quiz	Assignment 1 (programming) announced Aug 31
	29 Aug	1. Introduction	
	31 Aug	2. C++ review: Templates and deep copy; Up to sec 1.6 (excluding 1.3 and 1.5)	
2	Recitation	Discuss assignment 1, compilation, and makefiles.	None.
	5 Sep	3. STL, containers and iterators; Sec 1.7.	
	7 Sep	4. Vectors; Sec 1.8	
3	Recitation	Using a debugger, performance analysis	Assignment 1 due Sep 15
	12 Sep	5. Complexity analysis; Sec 2.1, 2.2, 2.6, and 2.7	
	14 Sep	6. Complexity analysis; Sec 2.3 - 2.5	
4	Recitation	Discuss assignment 2	Assignment 2 (programming) announced Sep 19
	19 Sep	7. Linked lists -- Up to sec 3.2	
	21 Sep	8. Self-organizing lists, STL lists; sec 3.5, 3.7	
5	Recitation	Discuss assignment 2	Assignment 2 due Sep 29 Assignment 3 (written) announced Sep 29
	26 Sep	9. Complexity analysis; Sec 2.8 - 2.9	
	28 Sep	10. 8. Complexity analysis using integration -- class notes	
6	Recitation	Discuss assignment 3	Assignment 3 due Oct 6 Assignment 4 (programming) announced Oct 6
	3 Oct	11. Deques; sec 3.8	
	5 Oct	12. Stacks and queues; Up to sec 4.3.	
7	Recitation	Discuss assignment 4	None.

	10 Oct	13. Stacks and queues in STL; sec 4.4 - 4.6	
	12 Oct	14. Midterm review	
8	Recitation	Midterm review	None
	17 Oct	15. **Midterm**	
	19 Oct	16. Recursion; up to sec 5.8	
9	Recitation	Discuss assignment 4 and midterm	None
	24 Oct	17. Iterators	
	26 Oct	18. Binary trees, searching, and traversal; up to sec 6.4, except sec 6.4.3.	
10	Recitation	Discuss assignment 5	Assignment 4 due Nov 3 Assignment 5 (programming) announced Nov 3
	31 Oct	19. BST insert and delete; sec 6.5-6.6	
	2 Nov	20. AVL trees -- insertion; sec 6.7 introduction and 6.7.2 (except deletion)	
11	Recitation	Discuss assignment 5	None
	7 Nov	21. AVL trees -- deletion; sec 6.7.2, and Self-adjusting trees; sec 6.8	
	9 Nov	22. Heaps; sec 6.9, except sec 6.9.2	
12	Recitation	Discuss assignment 6	Assignment 5 due Nov 13 Assignment 6 (written) announced Nov 13
	14 Nov	23. Heap initialization; sec 6.9.2 and class notes.	
	16 Nov	24. Proving properties of trees, and algorithms on them; class notes.	
13	Recitation	Discuss assignment 6.	Assignment 6 due Nov 22
	21 Nov	25. Hashing; up to sec 10.2.1	
	23 Nov	Thanksgiving -- no class	
14	Recitation	Lecture review.	None.
	28 Nov	26. Hashing; sec 10.2.2, sec 10.3	
	30 Nov	27. Graphs; up to sec 8.2	
15	Recitation	Finals review	None
	5 Dec	Other topics	
	7 Dec	28. Finals review	

16	11 Dec	**7:30 am - 9:30 am, Final exam (for section 1-2)**	
	13 Dec	**7:30 am - 9:30 am, Final exam (for section 3-4)**	

Grading criteria:

The overall grade for COP 4530 is an average of two equally weighted parts: (i) Exams and class participation, and (ii) Assignments. Exams consist of a midterm and a final exam. Class participation consists of quizzes, answering questions correctly in class, and other positive contributions, discussed in greater detail at www.cs.fsu.edu/~asriniva/courses/DS05/classparticipation.html. Assignments consist of four programming projects and two written assignments.

There are 1000 total points that may be earned in the course, distributed as shown in Table 1. You must earn at least 350 exam + class participation points, and 350 assignment points, to get a course grade of C or better. Once meeting this constraint, the final grade is determined using Table 2.

Table 1: Course Points

Item	*Points/Item*	*Item Total*	*Total*
Quizzes and Class Participation	100	100	
Midterm	150	150	
Final Exam (comprehensive)	250	250	500
Assignment 1, 3	50	100	
Assignment 2, 4, 5, 6	100	400	1000

Table 2: Letter Grades

Points	*Grade*
920 - 1000	A
900 - 919	A-
880 - 899	B+
820 - 879	B
800 - 819	B-
780 - 799	C+
720 - 779	C
700 - 719	C-
680 - 699	D+
620 - 679	D
600 - 619	D-
0 - 599	F

NOTE: You must earn at least 350 points in both components: (i) Exams and class participation, and (ii) Assignments, to be awarded a course grade of C or better. For example, if you obtain 500 points on the assignments, but only 300 in the rest, then you will not get a B-. Instead, you will get a C-, since that is the highest grade for which you will be eligible without obtaining 350 points in each of the two components, assignments and exam+class participation.

Programming assignment Assessment
Programming assignments will be assessed using Table 3 as a rough guide.

Table 3: Programming assignment assessment criteria	
Criterion	*Percentage Points Range*
Project compiles and executes simple sample inputs correctly	0 ... 20
Baseline test passed	0 ... 20
Advanced test passed	0 ... 20
Project meets requirements	0 ... 20
Good design and readability	0 ... 10
Project log and project submitted as specified	0 ... 10

The first three criteria will be assessed objectively through automated testing. A member of the instructional staff will then assess for the latter three criteria. Please note carefully the following important items:

- You must understand your project work. If you are asked to explain your work, and you are unable to do so, you may be assigned a grade of zero.
- In group projects, about half your score will be based on the work of the entire group, and about half the points will be for your contribution. You should, therefore, be able to clearly identify your contribution.
- The above grading guidelines are for projects that are reasonably correct. For example, if your code is very readable but does not have anything to with the problem you were asked to solve, then you will not get any points!

Course policies:

Attendance Policy:

The university requires attendance in all classes, and it is also important to your learning. The attendance record may be provided to deans who request it. If your grade is just a little below the cutoff for a higher grade, your attendance will be one of the factors that we consider, in deciding whether to "bump" you up to the higher grade. Missing three or fewer lectures will be considered good attendance. In rare cases, such as medical needs, religious holidays or jury duty, absences may be excused with appropriate documentation. You should let me know in advance, when possible, and submit the documentation I seek. You should make up for any materials missed due to absences.

Missed exam Policy:

A missed exam will be recorded as a grade of zero. We will follow the university rules regarding missed final exams (see

http://registrar.fsu.edu/dir_class/fall/exam_schedule.htm), for all the exams, including the final exam.

Late Assignment Policy:

In order to enable us to provide timely solutions to assignments, we have the following policy regarding submission of late assignments (unless related to an excused absence).

- An assignment that is turned in no more than 48 hours late will be scored with a 20% penalty.
- An assignment that is turned in more than 48 hours late will receive the score of zero, though we will review it and comment on it.

Grade of "I" Policy:

The grade of "I" will be assigned only under the following exceptional circumstances:

- The final exam is missed with an accepted excuse for the absence. In this case, the final exam must be made up during the first two weeks of the following semester.
- Due to an extended illness or other extraordinary circumstance, with appropriate documentation, the student is unable to participate in class for an extended period. In this case, arrangements must be made to make up the missed portion of the course prior to the end of the next semester.

Professional ethics:

You will gain confidence in your ability to design and implement algorithms only when you write the code yourself. On the other hand, one does learn a lot through discussions with ones peers. In order to balance these two goals, I give below a list of things that you may, and may not, do.

Things you may not do: You should not copy code from others. This includes directly copying the files, replacing variable names in their code with different names, altering indentation, or making other modifications to others' code, and submitting it as your own. (You may also wish to note that many of the modifications that make codes look very different in a higher level language, yield lower level representations that are very close, and hence easy to detect.) Furthermore, you should take steps to ensure that others cannot copy code from you -- in particular, you should have all permissions on assignment files and directories set off for others.

Things you may do: You may discuss specific problems related to use of the computer, useful utilities, and some good programming practices, with others. For example, you may ask others about how to submit your homework, or how to use the debugger or text editor. **Honor Code:** Students are expected to uphold the academic honor code published in "The Florida State University Bulletin" and the "Student Handbook". Please read the

provisions of the Academic Honor Code: http://www.fsu.edu/Books/Student-Handbook/codes/honor.html. Also read the section on "Honor code" below.

Plagiarism:

> Plagiarism is "representing another's work or any part thereof, be it published or unpublished, as ones own. For example, plagiarism includes failure to use quotation marks or other conventional markings around material quoted from any source" (Florida State University General Bulletin 1998-1999, p. 69). Failure to document material properly, that is, to indicate that the material came from another source, is also considered a form of plagiarism. Copying someone else's program, and turning it in as if it were your own work, is also considered plagiarism.
> **ADA:** Students with disabilities needing academic accommodation should (1) register with and provide documentation to the Student Disability Center, and (2) bring a letter to the instructor indicating the need for accommodation and what type. This should be done during the first week of class. For more information about services available to FSU students with disabilities, contact:
>
> Student Disability Resource Center
> Dean of Students Department
> 97 Woodward Avenue, South
> Florida State University
> Tallahassee, FL 32306-4167
> (850) 644-9566 (voice)
> (850) 644-8504 (TDD)
> sdrc@admin.fsu.edu
> http://www.fsu.edu/~staffair/dean/StudentDisability/
>
> This syllabus and other class materials will be made available in alternative format upon request.

SYLLABUS CHANGE POLICY:

This syllabus is a guide for the course and is subject to change with advanced notice.

Last modified: 20 Sep 2005