Department of Electrical and Computer Engineering
University of Puerto Rico
Mayagüez Campus

Syllabus for ICOM 4035 – Data Structures (CS2)
Fall 2002

1. Faculty

Dr. Manuel Rodríguez Martínez
Office: T-212
Phone: (787)-832-4040, x-3023
E-mail: manuelr@acm.org
Office hours: TBA, or by appointment

2. Teaching Assistant

TBA
Office: TBA
Phone: TBA
E-mail: TBA
Office hours: TBA, or by appointment

3. Course Description

Introduction to the design, analysis and implementation of data structures and sorting algorithms, using object-oriented programming techniques. Study of computational complexity and Big-O notation. Design and implementation of abstract data types and containers classes: Vector, Bag, List, Set, Sequence, Map, Stack, Queue, Priority Queue, Tree, and Graph. The list of concrete structures to be studied includes: dynamic arrays, linked lists, Bit vectors, binary trees, binary search trees, B-trees, heaps, hash tables, adjacency matrices and adjacency lists. Recursion, pointers, templates and inheritance will be used extensively. This course is often called **Computer Science II (CS2)** in many prominent U.S. universities.

4. Pre-requisites

ICOM 4015 or equivalent. Proficiency with C++ and UNIX.

5. Time and Place

Lecture: Tuesdays and Thursdays, 4:30 PM – 5:50 PM, S-230

Laboratory:
Section 166: Wednesdays, 12:30 PM – 3:20 PM, S-121
Section 167: Fridays, 12:30 PM – 3:20 PM, S-121

6. Credits

3 credits

7. Class Web Page

http://www.ece.uprm.edu/~manuel/class/fall02/icom4035/

You are responsible to read this Web page periodically to obtain class materials, and other important announcements about this class

8. Textbooks

Required:

Data Structures and Other Objects Using C++, 2nd Ed.
Michael Main and Walter Savitch
Addison-Wesley, 2000
ISBN: 0-201-70297-5

Recommended:

Essential C++
Stanley B. Lippman
Addison-Wesley, 2000
ISBN: 0-201-48518-4

C++ Primer, 3rd Ed.
Stanley B. Lippman and Josée Lajoie
Addison-Wesley, 1998
ISBN: 0-201-82470-1

9. Grading

Your grade will be based **exclusively** on the scores that you obtain in the class projects, exams and laboratory assignments. The curve to be used to assign a grade to your score will be as follows:

Score	Grade
100 – 90	A
89 – 80	B
79 – 70	C
69 – 65	D
64 – 0	F

Your total score will be calculated from your individual scores in the projects, exams and laboratory assignments. The weights assigned to each of these categories are as follows:

Programming Projects (4)	35%
Laboratory Work	10%
Midterm Exams (3)	35%
Final Exam (Comprehensive)	20%

There will be no special project, no special homework, no special exam, nor any other kind of "*special work*" to improve grades. However, each project or exam might have an extra credit problem that you can use to help improve your score in that corresponding category.

10. Exams

In this course, there will be three midterm exams and a comprehensive final exam. Unless otherwise indicated, all exams will be taken with closed books and closed notes. The midterm exams will be administered outside the regular class time. The date and time for each midterm exam will be as follows:

Exam Number	Date	Time	Place
I	September 18, 2002	6:00 PM – 8:00 PM	S-113
II	October 16, 2002	6:00 PM – 8:00 PM	S-113
III	November 25, 2002	6:00 PM – 8:00 PM	S-113

The final exam will be administered in accordance with the schedule specified by the Registrar of the University of Puerto Rico, Mayagüez Campus.

The lowest score in the midterm exams can be replaced with the score in the final exam, provided that the score in the final exam is higher. Otherwise, the scores in the midterm exams will neither be replaced nor dropped.

Each question included in each exam (midterm or final) will fall into one of the following categories:

- Explanation of a technical concept.
- Proof of a mathematical proposition.
- Solution to a problem using the concepts discussed in class.
- Tracing of either C++ code segments or algorithms.
- Implementation of C++ classes and code segments.

10.1 Exam Reposition Policy

In this course, there will be **NO** repositions for missed midterm exams. If a student misses one midterm exam then that scored will be replaced with the score obtained in the final exam. Any other missed midterm exam will carry a score of 0.

11. Incomplete Grade Policy

A student will receive an incomplete grade if and only if the student misses the final exam and has a valid excuse. Such excuse must be one of the following:

- Medical certificate indicating illness.
- Legal certificate indicating an appointment to attend a Court of Law.
- Certificate from a hospital or a physician indicating the death of either: parent, child, husband, wife or sibling.

12. Programming Projects

In this course, you are expected to complete four programming projects that are designed with the following objectives**:**

1) Test your knowledge of the data structures presented in class.
2) Test your individual skills for engineering a programming solution to a particular problem.
3) Provide experience in the design and implementation of complex software modules using object-oriented techniques.

You will be given **three weeks** to complete each programming project. You must implement your project using the C++ programming language, and you must work individually. You might discuss with your peers general aspects about the project and/or programming environment. However, you cannot share your code with any student, nor use code written by someone else. Failure to comply with this requirement will be considered as an act of academic dishonesty and you will receive a grade of F in the class (**read section below titled Academic Integrity**).

You must submit your project electronically following a procedure that will be discussed in class. For each project, you will be given a **tar file** containing a directory, called the *project directory*, with the following items:

1) A document explaining the tasks to be completed for the programming project.
2) A document indicating a minimal set of operations that your program must execute to be considered a *running program*. **If your program neither compiles nor performs this minimal set of operations it will receive a score of 0.**
3) A *make* file with the commands needed to compile the modules in the project.
4) A set of .h files containing the declarations of the classes and methods to be implemented in the project. **NOTE**: For project number 4 you must create these .h files.
5) A set of .cpp files containing **empty** implementations of the methods associated with the classes, algorithms and other tasks related with the particular programming project. **It is your job to implement the C++ code that executes the tasks these methods are designed to perform.**
6) A set of test input files and their corresponding test output files. You should use these to help you decide whether your program is working correctly or not, based on what type of output your program produces out of these test input files. **NOTE**: This set of input files will not be the only one to be used to grade your project. Hence, a program might pass all the tests in these test files, and yet fail some of the extra tests used for grading. However, if your program executes correctly on the test input files you will receive at least 70% of the total score for a given project.

Your project directory should contain all the files associated with your project. Once you have completed your project, you will create a **new** tar file that **must** contain everything that you have created in your project directory. You will submit this tar file for us to grade your project. Again, you will receive further instructions on how to submit your project electronically.

You are expected to work in the Amadeus UNIX lab provided by the University of Puerto Rico, Mayagüez Campus. But, you are free to use your own UNIX computer, if you prefer to do so. **However, you must ensure that the programs that you submit for grading do compile and execute correctly**

on the machines available in the Amadeus UNIX lab, since the projects will be graded there. Failure to comply with this requirement will result in a score of 0 for the project being evaluated. **NOTE: This policy will be strictly enforced**.

Late Project Policy:

Each project will have a due date composed of an hour, month and day (i.e. 4:00 PM-September 12). A project will be considered late if it is submitted for grading one minute after its due date. For example, if the due date for a project is 3:00 PM-October 31, then a project submitted at 3:01 PM-October 31 is considered as one day late. Any late project will receive the following penalty:

1 day past due date	-15%
2 days past due date	-30%

No project will be accepted for grading if submitted 3 or more days after its due date, and any such project will receive a score of 0. Any project that is not submitted for grading will automatically receive a score of 0.

NOTE: We will not debug your code via e-mail. We shall only look at your program source code listings, or login to see your code files during the allotted office hours.

13. Laboratory

In this course there will be a weekly 3-hour laboratory session in which the Teaching Assistant will further discuss the concepts presented in class, and the programming projects. There will be several problem sets, and various small programming exercises as part of the assignments associated with the laboratory. Information regarding each laboratory assignment will be given well in advance of the laboratory session.

14. Academic Integrity

Each student is expected to work individually on all projects, exams and laboratory assignments. You may not share your answers to the laboratory assignments. You may not use code from another student, or code that you find on the Internet or any similar resources. You may not share your code with another student. Failure to comply with these requirements will result in a grade of F in the course for the student(s) breaking these rules. Unauthorized group efforts, particularly during exams, will be considered academic dishonesty and the students involved will receive an F in the course. You should read Article 10 of the “Reglamento General de Estudiantes de la Universidad de Puerto Rico” to learn more about the possible sanctions that you might experience if caught in an act of academic dishonesty.

15. List of Topics

The following is a list of the course topics in the order in which they will be presented. This list is subject to change and it will vary depending on the pace of the lectures.

TOPICS:

1. Discussion of the Course Syllabus
2. The Phases of Software Development

    a. Specification, Design and Implementation
    b. Testing and Debugging
3. Object-Oriented Programming Using C++
    a. Classes and Members
    b. Constructors
    c. Namespaces, Header Files and Implementation Files
    d. Parameter Passing
    e. Operator Overloading\
4. Derived Classes and Inheritance
    a. Derived Classes
    b. Virtual Functions
5. Computational Complexity
    a. Big-O Notation
    b. Worst-Case Analysis
    c. Average-Case Analysis
    d. Best-Case Analysis
6. Container Classes
    a. Bag Container Class
    b. Sequence Container Class
    c. Set Container Class
7. Pointers and Dynamic Arrays
    a. Pointers
    b. Dynamic Arrays
    c. Vector Container Class
    d. Bag Container Class with a Dynamic Array
8. Linked Lists
    a. Singly-Linked Lists
    b. Doubly-Linked Lists
    c. Circular-Linked Lists
    d. Bag, Set and Sequence Container Classes using Linked Lists
9. Templates
    a. Template Functions
    b. Template Classes
    c. C++ Standard Template Library (STL)

10. Class Iterators
    a. STL Iterators
    b. Linked List Iterator
    c. Bag Container Class with a Linked List and an Iterator
11. Stacks
    a. Stack Interface and Applications
    b. Dynamic Array Implementation
    c. Linked List Implementation
12. Queues
    a. Queue Interface and Applications
    b. Dynamic Array Implementation
    c. Linked List Implementation
    d. Priority Queues
13. Recursion
14. Trees
    a. Binary and General Trees
    b. Array Representation
    c. Pointer Representation
    d. Tree Traversals
        i. Pre-order
        ii. In-order
        iii. Post-order
15. Binary Search Trees
16. Heaps and Priority Queues
17. B-trees
18. Maps and Searching
    a. Serial Search
    b. Binary Search
    c. Open-Address Hashing
    d. Chained Hashing

19. Graphs
    a. Undirected Graphs
    b. Directed Graphs
    c. Adjacency Matrix Implementation
    d. Adjacency List Implementation
    e. Graph Traversals