**ESSEX COUNTY COLLEGE**

**Engineering Technologies and Computer Sciences Division**

**CSC 225 – Data Structures**

**Course Outline**

**Course Number & Name:** CSC 225 Data Structures

**Credit Hours:**  4.0 **Contact Hours:**  4.0 **Lecture:** 4.0 **Lab:**  N/A **Other:**  N/A

**Prerequisites**:  Grade of “C” or better in CSC 122

**Co-requisites:** None **Concurrent Courses:** None

**Course Outline Revision Date:**  Fall 2010

**Course Description**: This course is designed to present the fundamentals of data structures from an object-oriented perspective. The course introduces students to the design and implementation of abstract data types using an object-oriented programming language. The course includes introduction to algorithm analysis, recursion, and internal and external sorting/searching methods. The fundamental concepts of inheritance and virtual functions are also examined. Students are required to complete a series of programming projects that demonstrate their understanding of lecture topics.

**Course Goals:** Upon successful completion of this course, students should be able to do the following:

1. construct object-oriented applications using ADT;

1. design applications, which implement various searching and sorting algorithms;
2. construct dynamic and static data structures within an application; and
3. apply the basic principles of object-oriented programming.

**Measurable Course Performance Objectives (MPOs)**: Upon successful completion of this course, students should specifically be able to do the following:

1. Construct object-oriented applications using ADT:

* 1. *define ADT and other related concepts*;
	2. *explain and utilize the basic structure of programming language;* and
	3. *construct programs using ADTs implemented with class structures (both user-designed as well as available in the programming library)*

2. Design applications, which implement various searching and sorting algorithms:

2.1 *design applications, which implement various searching and sorting algorithms;* and

2.2 *use programs to test the efficiency and correctness of the searching and sorting algorithms*

**Measurable Course Performance Objectives (MPOs)** (continued):

3. Construct dynamic and static data structures within an application:

3.1 *distinguish between dynamic and static data allocation;*

3.2 *compare the relative merits of the dynamic and static data systems;*

3.3 *create the source code that implements dynamic and static data structures*

4. Apply the basic principles of object-oriented programming:

4.1 *document the source code so the programs are self-explanatory;*

4.2 *explain a problem using the source code;* and

4.3 *define and use appropriate object-oriented program terminology in written and oral form*

**Methods of Instruction**: Instruction will consist of lectures, laboratory demonstrations and assignments, and programming examples.

**Outcomes Assessment:** Exam questions are blueprinted to course objectives. Checklist rubrics are used to evaluate the programming projects for the presence of course objectives. Data is collected and analyzed to determine the level of student performance on these assessment instruments in regards to meeting course objectives. The results of this data analysis are used to guide necessary pedagogical and/or curricular revisions.

**Course Requirements:** All students are required to:

1. Maintain regular attendance and take part in class discussions.

2. Complete assigned homework and programming projects on time.

3. Take all exams as scheduled.

**Methods of Evaluation:** Final course grades will be computed as follows:

 **% of**

**Grading Components final course grade**

* **Homework, class participation and attendance 10%**

Students must practice skills on their own by doing homework to be able to master course objectives. Homework assignments relate to these objectives**.** Attendance and class participation are necessary for students to benefit from the guidance of the instructor.

* **10 or more programming projects** (dates specified by the instructor) **35%**

Programming projects will show evidence of the extent to which students meet course objectives. Students should show that they have synthesized a combination of concepts. (See page 6 for programming project guidelines.)

**Methods of Evaluation** (continued)**:**

 **% of**

**Grading Components final course grade**

* 2 or more Exam s 25%

Exams will provide evidence of the extent to which students have mastered course objectives.

* **Final Exam**  **30%**

The same objectives apply as with exams, but it is anticipated that students will provide increased evidence of synthesizing a combination of concepts.

**Academic Integrity:** Dishonesty disrupts the search for truth that is inherent in the learning process and so devalues the purpose and the mission of the College. Academic dishonesty includes, but is not limited to, the following:

* plagiarism – the failure to acknowledge another writer’s words or ideas or to give proper credit to sources of information;
* cheating – knowingly obtaining or giving unauthorized information on any test/exam or any other academic assignment;
* interference – any interruption of the academic process that prevents others from the proper engagement in learning or teaching; and
* fraud – any act or instance of willful deceit or trickery.

Violations of academic integrity will be dealt with by imposing appropriate sanctions. Sanctions for acts of academic dishonesty could include the resubmission of an assignment, failure of the test/exam, failure in the course, probation, suspension from the College, and even expulsion from the College.

**Student Code of Conduct:** All students are expected to conduct themselves as responsible and considerate adults who respect the rights of others. Disruptive behavior will not be tolerated. All students are also expected to attend and be on time for all class meetings. No cell phones or similar electronic devices are permitted in class. Please refer to the Essex County College student handbook, *Lifeline*, for more specific information about the College’s Code of Conduct and attendance requirements.

**Course Content Outline:** based on the text *Data Structures with C++ Using STL*, 2nd edition, by William Ford and William Topp; published by Prentice Hall, 2002; ISBN #: 0-13-085850-1

Additional required course materials: USB flash memory and a note book for taking notes.

**Week**

**(3 meetings**

**@ 80 minutes) Topic/Chapter**

1 – 2 Introduction to Data Structures and Algorithms (Chapter 1)

1. Abstract Data Types
2. Implementing C++ Classes
3. Declaring and Using Objects
4. Application Programming Interface (API)
5. Strings

3 – 4 Object Design Techniques (Chapter 2)

1. Software Design
2. Object Composition
3. Operator Overloading

**First Examination**

5 – 6 Introduction to Algorithms (Chapter 3)

1. Selection Sort
2. Simple Search Algorithms
3. Analysis of Algorithms
4. Analyzing Search Algorithms

7 – 8 Vector Container (Chapter 4)

1. STL Container Classes
2. Template Classes
3. Vector Class
4. Vector Applications

9 – 10 List Container (Chapter 6)

1. List Container
2. Iterators
3. List Insert and Erase Operations

 **Second Examination**

11 Stacks (Chapter 7)

Stack ADT

Stack Implementation

Postfix Evaluation

**Week**

**(3 meetings**

**@ 80 minutes) Topic/Chapter**

12 Queues and Priority Queues (Chapter 8)

1. Queue ADT
2. Queue Class
3. Priority Queue

13 Linked Lists (Chapter 9)

1. Linked List Nodes
2. Building Linked Lists
3. Implementing Linked Queue
4. Doubly Linked List

14 Trees (Chapter 10)

1. Tree Structures
2. Binary Tree Nodes
3. Binary Tree Scan Algorithms
4. Binary Search Trees

15 Advanced Associative Structures (Chapter 12)

1. Hashing
2. Designing Hash Functions
3. Hash Class
4. Hash Table Performance

 **Final Examination**

Note: 10 or more programming projects are assigned on an ongoing basis throughout the semester to correspond to the topics being discussed in class. Roughly one programming project is due each week (with the exception of exam weeks).

**CSC 225 Project Guidelines**

1. All programs must be documented. The student’s name must appear at the top of the source code and every other file.
2. The project number and the date the project is due and the date it is handed in must be at the top of the documentation for the source code.
3. The student must explain what the program does, how it does it and what the known limitations are (if any exist).
4. In cases where there are library files, these should also be documented.
5. The C++ code should also be documented. The student should have a comment for every two to three lines of code.
6. If a student decides to complete a program that is different from the given assignment, the student must explain that in the documentation of the program.