**Data Structures and Algorithms I Syllabus**

**CMP 338: Data Structures and Algorithms I.** 4 hours, 4 credits. Abstract characterizations as well as the design and implementation of data structures such as arrays, stacks, queues, linked lists, binary search trees, heaps, and graphs along with algorithms that make use of such structures including algorithms for sorting, searching, and memory management, will be studied. Algorithms will be analyzed for their asymptotic behavior in terms of time complexity and space requirements will be considered as well. Implementation issues will be considered and students will write programs that embody these data structures and algorithms.

**Prerequisites:**

**CMP 326** – In this course we will do extensive programming in Java. It is assumed that all students are capable of reading and writing object oriented Java code.

**CMP 232** – In this course we will demonstrate correctness and efficiency of many of the algorithms presented via mathematical arguments and proofs. It is assumed that students are capable of following the logic of such when presented.

**Professor:** Brian Murphy

**Office:** GI-211C

**Phone:** (718) 960-5117

**Email:** brian.murphy@lehman.cuny.edu

**Grading Policy**

**Expectations:** Students are expected to learn the material covered in class, from the textbook, and any other assigned reading. Completing programming projects is a very important part of the learning experience. Students must write readable and complete programs that execute correctly. Students should review material from prior courses as needed using old notes and textbooks.

**Assignments:** In addition to explicitly assigned problems, students should implement every data structure and algorithm that is covered in class.

**Exams:** There will be a midterm and a final exam.

**Grades:** The precise grading policy will be explained in class by your instructor.

**Materials, Resources and Accommodating Disabilities**

**Textbook:** Data Abstraction and Problem Solving with Java: Walls and Mirrors by Frank M. Carrano and Janet J. Prichard

**Technology:** Students will need to have access to personal computers with Java IDE software installed. Such computers are available for student use on campus. For students with computers at home, Java IDE Software is available free of charge on the internet. Speak to your instructor for details.

**Accommodating Disabilities:** Lehman College is committed to providing access to all programs and curricula to all students. Students with disabilities who may need classroom accommodations are encouraged to register with the Office of Student Disability Services. For more information, please contact the Office of Student Disabilities, Shuster Hall, Room 238, phone number (718) 960-8441.

**Course Objectives**

At the end of the course students should:

1. Have improved object-oriented programming skills.
2. Have improving understanding of recursive methods.
3. Understand a core group of basic data structures as enumerated in topics below.
4. Be able to conceptualize many programming issues at a higher level through data structures.
5. Know the tradeoffs of each studied data structure so as to employ the appropriate one for a given situation.
6. Be able to incorporate data structures for efficient handling of data into their programs.
7. Be able to code algorithms involving data structures using an object oriented programming language.
8. Be able to analyze data structures and their algorithms for asymptotic behavior.
9. Achieve a level of maturity in the subject so that further study of data structures can be pursued independently.

**Review Topics:**

Recursion

Searching: Linear Search versus Binary Search

Sorting: Bubble Sort, Insertion Sort, Selection Sort, Quick Sort

**Topics:**

Abstract Data Types (ADTs)

Specification and Implementation

Asymptotic Analysis and Notation: “Big-Oh”

Sorting: Merge Sort, Heap Sort, Counting Sort, Radix Sort

Stacks

The Stack ADT

Application: Evaluation of Arithmetic Expressions

Array Implementations: Fixed size and resizable.

Linked List Implementation

Comparisons of efficiency for Array and Linked List implementations under various scenarios

Queues

The Queue ADT

Array Implementations: Fixed size and resizable and their shortcomings

Circular Array Implementation

Linked List Implementation

Linked Lists

A simple List ADT

Implementing a List

Implementation Issues

The use of dummy nodes

Binary Search Trees

Definitions for Binary Tree and Binary Search Tree

Implementing Binary Trees using Linked Nodes

Implementing a List

Properties of Binary Trees

Full, Balanced, Complete Binary Trees

Additional methods for manipulating Binary Tree Data

Using the definitions to determine correctness of Binary Tree Algorithms

Implementing Binary Trees using Arrays

Heaps

Definitions of Max-Heaps and Min-Heaps

Implementing a Heap

Using a Heap to Implement Heapsort

Graphs

Graph ADT and Definitions

Data Structures and Implementation issues for Graphs

Graph Traversal: Breadth First and Depth First Traversal

Greedy Algorithms

Shortest Path: Dijkstra’s Algorithm

For each algorithm, verification of its correctness and analysis of its efficiency will be considered.