## Department Master Syllabus

## Camden County College

## Blackwood, New Jersey

**Course Title: Computer Science II**

**Course Number: CSC-223**

## Department/ Program Affiliation: Computer Science

**Date of Review:**

(This Department Master Syllabus has been examined by the program/department faculty members and it is decided that no revision is necessary at this time.)

**Date of Last Revision**: October 2023

(This Department Master Syllabus has been examined by the program/department faculty members and it is decided a change requiring a revision is necessary at this time.)

**N.B.** A change to the course materials alone (textbooks and/or supplementary materials) may not constitute a revision. Any other change to the items listed below on this form is considered a revision and requires approval by the program faculty at a Program/Department Meeting and by the division at a Chairs and Coordinator Meeting.

## Credits: 4

## Contact Hours: Lecture 3 Lab 3 Other 0

**Pre-requisites: CSC-122 Computer Science I**

**Co-requisite: MTH-129**

**Course Description:** This course is the third in a three-course sequence for Computer Science majors and has a supervised, in-class computer programming laboratory component. The course expounds on recursion, analysis of algorithms (Big O), classic data structures, and the software lifecycle in order to lay a foundation for prevailing software engineering practices for large software systems as well as examining professionalism and ethics in the computing disciplines. The concepts of creating software solutions designed for high cohesion and low coupling are stressed through building reusable software components. Major topics include recursion and recursive algorithms, generics, lists, stacks, queues, trees, BSTs, heaps, sets, graphs and hash tables. Programming solutions that use dynamic data structures and standard templates will be implemented as students apply the concepts above through individual, paired, and team programming assignments in an object-oriented language.

**Course Student Learning Outcomes:**

Upon completion of this course, the student will be able to:

* Examine the issues of large-scale software development projects including the ramifications of security, the importance of professionalism and ethical considerations of computing professionals as assessed by graded exercises, homework assignments, programming projects, and exams.
* Construct and utilize UML diagrams (class, use case and sequence) for analysis and design in addition to identifying and using common design patterns as assessed by graded exercises, homework assignments, programming projects, and exams.
* Analyze various searching and sorting algorithms in terms of efficiency using Big O and distinguish when each provides a preferred approach as a solution as assessed by graded exercises, homework assignments, programming projects, and exams.
* Distinguish among and incorporate the use of standard class libraries (of pre-built data structures and iterators) to develop best-fit software solutions to problems as assessed by graded exercises, homework assignments, programming projects, and exams.
* Design and implement programming solutions with various data structures including Trees, Queues, Stacks, Hash Tables and Lists by analyzing and choosing the appropriate data structure based on the program specification while weighing memory and efficiency requirements and tradeoffs as assessed by graded exercises, homework assignments, programming projects, and exams.
* Analyze recursive algorithms and write recursive programming solutions where appropriate as assessed by graded exercises, homework assignments, programming projects, and exams.
* Compare and use directed and undirected graphs as Adjacency matrices and/or Adjacency lists in programming solutions; tracing transitive closure, topsort, breadth-first and depth-first algorithms as assessed by graded exercises, homework assignments, programming projects, and exams.

**General Education Student Learning Outcomes (if applicable):**

**N/A**

**Course Outline:**

1. Course Introduction, SW Engineering principles, Software Lifecycle
2. Software Reliability, Program Correctness, Professionalism and Ethics
3. Program Efficiency, analyzing searching and sorting algorithms using Big O
4. Using recursion, analyzing recursive algorithms
5. Lists (arrays and linked implementations), Stacks and Queues
6. Using pre-defined standard templates(Collections Framework) with iterators
7. Tables and Hashing
8. Trees, BST, Heaps, and Priority Queues
9. Graphs
10. Sets and Maps

**Course Activities:**

The classroom activities will include formal and informal lectures and lab sessions. During lectures, new material and concepts are explained and demonstrated. Students are encouraged to contribute to the discussion and to ask questions about the material. Structured, supervised, "hands-on" lab sessions are built-in to the semester schedule. These sessions are structured to amplify lecture material. Attendance during lab time and completion of the laboratory exercises is a crucial component to mastering the course material. Most laboratory exercises are paired-programming or conducted as small team exercises, therefore your attendance and full participation is required. Some class time may be allotted to start required projects in class, but projects are expected to be completed outside of the regularly scheduled labs and class meetings.

**Assessment of Student Learning Outcomes:**

The student will be evaluated on the degree to which student learning outcomes are achieved. A variety of methods may be used such as tests, class participation, projects, homework assignments, etc. (There must be some evidence that the learning outcomes have been achieved.)

**Course Materials:**

**Textbook(s):** To be determined

**Supplemental Materials:** Announced first day of class as needed.