CSCI 106 Data Structures

SEMESTER (Spring 2017)

PROFESSOR/CLASS INFORMATION

Grace Lew

(Course) Title: Data Structures
Term: Spring, 2017
Class Location: LIB 141
E-Mail: grace.lew@biola.edu
Office Phone: 562 903-4741
Office Hours: See the announcement under the class website at **http://csci.biola.edu/csci106/**

Class Days/Time:
MW 10:30-11:45am (Section I)
MW 12:00- 1:15pm (Section II)
Credit Hours/Units: 3
Office Location: Grove 8
Meetings with Professor: Make Appt via Email
Admin Assistant: Jerrianne Smith, x4741

COURSE DESCRIPTION

Linear lists, strings, arrays and orthogonal lists; graphs, trees, binary trees, multi-linked structures, searching and sorting techniques, dynamic storage allocation; applications. Offered every year

COURSE ALIGNMENT WITH PROGRAM LEARNING OUTCOMES

CSCI 106 Data Structures: This lower-division course is a core course required of all Computer Science majors designed to be taken within the first year of the program. Successful completion of this course (see next section) will prepare students to demonstrate a developing proficiency toward the accomplishment of PLO: programming and system integration.

COURSE OBJECTIVES AND STUDENT LEARNING OUTCOMES

By the completion of this course including class participation, class assignments (referred to as "Tasks"), class readings and group interaction, the following objectives and learning outcomes will be assessed and demonstrated:

IDEA Objective #4: Developing specific skills, competencies, and points of view needed by professionals in the field most closely related to this course (Essential emphasis).

STUDENT LEARNING OUTCOMES (The learner will demonstrate that he or she has satisfactorily fulfilled IDEA Objective #4 by being able to):

- use the fundamental data structures in computer science, such as vectors, linked lists, and search trees in programming assignments
- write programs based on the fundamental concepts of object-oriented programming and the skills of designing and implementing data structures and abstract data types as C++ classes, and
- conduct basic software-engineering practice in the design, implementation, and testing of programs.

REQUIRED TEXTS

Required Textbooks (Each of the following books are required and will be used in this course)

- Tony Gaddis, *Starting Out With C++: From Control Structures Through Objects*, 8th Ed., 2015.
- Stanley Lippman, etc. *C++ Primer (5th Ed)*, Addison Wesley, 2013.
- **Keep *C++ Primer* if you are going to take CSCI 230 in fall. It will be one of the textbooks for CSCI 230.**

LEARNING TASKS (Assignments) & ASSESSMENT (Grading)

Description and Weighting of Assignments:

Task 1: Weekly Reading and Progress Report

Due Date: Monday of the week **(14 assignments)**
Weighting: 15%
Possible Points: 4 points for each assignment.
Description:
For each reading assignment, the student needs to finish the reading on time and submit the following information online as a report.

Effort (2 points):
Record the information such as (i) a numerical amount of time he/she spent for the reading, (ii) a numerical percentage regarding the percentage of stuff in the reading actually read and understood, and (iii) whether the student has come to the class this week.

Reflection on the reading (2 points):
The student need to put down 1 to 2 paragraphs of his/her thoughts about the reading such as new insight you gained, interesting things encountered, questions of things you don't understand, and so forth.

Assessment:
For the effort part,
the student is expected to **(i)** have attended the class this week at least once (0.5 point), and **(ii) have either** gained a good understanding of **80% or more of the contents or** have spent at least **three hours** in the reading (1.5 points).

For the reflection part,
the student is expected to show substantial evidence of understanding or effort of trying to understand the contents in the reading.

Task 2: Programming Assignments (9-11 assignments)

Due Date: Monday of the week
Weighting: 40%
Possible Points: 6 points each.

Description: There will be about 9 programming assignments, which form the backbone of the course. They require the student to incrementally develop programming skills based on the concepts of data structures and object-oriented programming learned in the class. You need to submit a peer review report together with all your source code files for each assignment **as a zip file**. In the self-evaluation report, you should describe results from sufficient test cases that are verified by a peer reviewer.

Integrity rules for programming assignments and programming tests:

- **Peer discussion is encouraged**: Peer discussion is encouraged to cultivate an open learning environment in the class, but you should carefully read the guidelines below to avoid any dishonest behavior and never step over the guidelines explicitly described in the following.
- **Never use code written by others**: Any copy-and-paste of code from other people's programs or from websites is viewed as cheating and you will get 0 points for the assignment.
- **Never circulate your code to others**: Peer discussion of code shown on the screen is acceptable for debugging purpose and for explanation of ideas. But you should never pass around your code (electronically or on paper) to others except for the TA and the instructor. Violating this rule is viewed as cheating in the class and the provider will receive 0 points for the assignment.
- **Never provide false or exaggerated results of test cases**: You need to describe results of test cases in the self-evaluation report**.** Providing false or exaggerated results of test cases in the report is viewed as cheating and you will receive 0 points for the assignment.
- **Demonstrate the credibility of your authorship of the work**: When you submit your code as your own work for points, you should make sure that you are able to explain your code and reconstruct your code from scratch without any outside help when requested. If you are not able to do that on your own when requested, you will get 0 points for the assignment and there will be an investigation.
- **Consequence of cheating in the class**: Cheatings end in 0 points for the assignments followed by discipline actions described in the student handbook.

Assessment: The student needs to submit (all related .cpp and .h files) together with a self-evaluation report. The self-evaluation report should describe results from sufficient test cases that are verified by a peer reviewer. We'll grade each programming assignment in a 0-6 scale based on the following rubric.

0. Nothing done or **missing the self-evaluation report** or **missing the integrity review** in the report.
1. Source code is completed but the code fails to compile successfully.
2. Source code can compile and do something required, but has serious bugs or miss a couple of key features.
3. Source code can compile and do most of the features required, but has many minor bugs or miss a key required feature.
4. Source code can compile and do all the features required, nearly fully functional, only a couple of minor bugs.
5. Source code can compile and do all the features required, fully functional, no bugs.
6. In addition to the points received above, get one more point if

   a. **the self-evaluation report contains sufficient descriptions of test cases used (0.25 point)**, and
   b. **the self-evaluation report indicates the results of the test cases were verified by a peer reviewer (0.25 point)**, and
   c. **the source code is well indented and commented to make it visually very readable (0.5 point)**.

Task 3: Exams (2-3 Tests)

Weighting: 45%

Description: Each test may have a written component, or a programming component, or both. The written component mainly tests the conceptual understanding of data structures while the programming component tests skills in object-oriented programming.

Assessment: The written component will be graded based on the answers provided while the programming component will be graded based on the same rubric provided for the weekly programming assignments.

CLASS INFORMATION

Late policy

- **No submission accepted after the submission site on Canvas is closed**: All submissions should be done through the Biola Canvas system. No submission will be accepted after the submission site on Canvas is closed, except for extremely exceptional situations such as a serious disabling health problem with evidence from the doctors.

- **Penalty for late submission after the due date but before the submission site is closed**:
  For a programming assignment,
  the submission site on Canvas may remain open for 2 more days after the due date and 1 point will be deducted from the work for a late submission before the submission is closed.
  For a reading report,

the submission site on Canvas may remain open for 2 more days after the due date and 1 point will be deducted from the report for a late submission before the submission is closed.

Computation of Final Grade:

Weekly Progress Report	15 %
Weekly Programming Assignments	40 %
Exams	45 %
Total	**100%**

Final grades will be awarded on the following point system:

A	93%
A-	90%
B+	87%
B	84%
B-	80%
C+	77%
C	74%
C-	70%
D+	67%
D	64%
D-	60% to pass class

Posting of Grades:

Grades for individual assignments will be posted under Biola Canvas system. To access the records online, log on to http://canvas.biola.edu/ to make sure the records are accurate.

Tentative Schedule

✧ Week 1	Objects and C++ Classes: data members + member functions
✧ Week 2	Objects and C++ Classes: operator overloading and friends
✧ Week 3	C++ vector class; Concept of recursion
✧ Week 4	Basic of pointers and dynamic memory allocation
✧ Week 5	More on pointers and dynamic memory allocation

Week 6	Functions, references, reference parameters
Week 7	*Midterm Review*
Weeks 8-9	Linked lists: implementation and application
Week 10-11	Binary Search Trees: implementation and application
Week 12	Variants of Search Trees; Heaps
Week 13	Stacks and queues: implementation and application
Week 14	Hash tables
Week 15	The C++ Standard template library (STL)
	Final