Programming Languages

CSCI 230

SEMESTER (fall 2016)

PROFESSOR/CLASS INFORMATION

Shieu-Hong Lin

(Course) Title: Programming Languages
Location: LIB 141
E-Mail: **shieu-hong.lin@biola.edu**
Office Hours: M to Th 8:30-10:30am
MW 11:30am~1:00pm
Meeting with Professor: **Make Appt via Email**
Office Location: Grove 8

Course Code/#: CSCI 230
Term: fall, 2016
Class Days/Time: T Th 12:00-1:15pm
Credit Hours/Units: 3
Office Phone: 562 903-4741
Dept. Website: http://csci.biola.edu
Admin Assistant: Jerrianne Smith, x4741

COURSE DESCRIPTION

Organization and structure of programming languages. Runtime behavior and requirements of programs. Introduction to programming language specifications and analysis. Study of various alternative languages such as C++, Java, and Python. Offered every year.

COURSE ALIGNMENT WITH PROGRAM LEARNING OUTCOMES

This lower-division course is a core course required of all Computer Science majors designed to be taken within the second year of the program. Successful completion of this course (see next section) will prepare students to demonstrate a developing proficiency toward the accomplishment of the PLOs: 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):

1. Develop the foundational understanding of key concepts such as context-free grammars, regular expressions, lexical and syntax analysis, interpreters and compilers and their roles in the design and implementation of the modern programming languages.

2. Cultivate the skills of problem solving, object-oriented programming, and basic software engineering using C++ through a semester-long programming project implementing an interpreter that can run programs for the very simple Biola programming language.

3. Develop an in-depth understanding of key features of object-oriented programming languages such as C++ and Java in areas such as memory management, data types and type checking, inheritance and polymorphism, templates and generic programming to effectively utilize them in programming tasks.

4. Cultivate the understanding of (i) scripting languages including Python and Java Script and (ii) markup languages such as HTML, XML, and their application in web programming by contrasting and relating the features between C++ and the new languages.

REQUIRED TEXTS

- Stanley Lippman, etc. *C++ Primer (5th Ed)*, Addison Wesley, 2012.
- R. Sebesta, *Programming the World Wide Web*, 8th ed. Pearson, 2014.

LEARNING TASKS (Assignments) & ASSESSMENT (Grading)

Description and Weighting of Assignments:

Task 1: Weekly Reading and Progress Report (15 assignments)

Weighting: 10%
Possible Points: 4 points each.
Description: The student needs to report the following information:

Effort (2 points):
Report the (i) a numerical estimate of the 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 (4 modules)

Weighting: 40%
Possible Points: 6 points each.
Description: There will be 4 modules to complete in the programming assignments, which flesh out a framework of an interpreter program. They require the student to incrementally develop programming skills based on the concepts of data structures and object-oriented programming learned in the class.

Integrity rules for programming assignments:

- **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.

1. Nothing done **or missing the self-evaluation report or missing the integrity review** in the report.
2. Source code is completed but the code fails to compile successfully.
3. Source code can compile and do something required, but has serious bugs or miss a couple of key features.
4. Source code can compile and do most of the features required, but has many minor bugs or miss a key required feature.
5. Source code can compile and do all the features required, nearly fully functional, only a couple of minor bugs.
6. Source code can compile and do all the features required, fully functional, no bugs.
7. 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: **Language Study Assignments**

Due Date: Distributed throughout the semester.
Weighting: 20%
Description: We'll assign a sequence of study assignments on key features in modern programming languages such as memory management, built-in data types, static typing or dynamic typing (type checking), control flow, subroutine call and parameter passing, and garbage collection, templates and generic programming, object-oriented programming (inheritance and sub-type polymorphism), functional programming, and so forth.
Assessment: The submitted work done will be examined to determine whether a satisfactory level of study has been done based on the breadth, depth, and clarity of the submitted work.

Task 4: Exams (Quizzes, midterm and final exams)

Due Date: Once each month for the quizzes, Exams in the midterm and final exam week
Weighting: 30%
Possible Points: Up to 50 points each.
Description: The exams have both the written component, which mainly tests the conceptual understanding of data structures, and the programming component, which 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

1. Class Attendance and Attendance Policy:

Attendance: You are expected to attend the class regularly since we will examine programming projects and various concepts of programming languages in the class. Missing the class may seriously hamper your understanding of many key concepts and programming skills critically needed in your programming assignments. Class attendance is counted toward points for the weekly progress report.

2. Turning in Assignments:

Assignments are expected to be electronically submitted under the Canvas system. Due dates are all on Tuesdays. The submission link under Canvas may remain open for 2 more days after the due date.

3. Late Policy:

1 point will be deducted for late submission within 2 days of the due date while the submission link is still open. **You will receive no points after the submission on canvas is closed** unless it is something like a serious health issue with statements from the doctor as proof.

4. Computation of Final Grade:

Weekly Progress Report	10 %
Weekly Programming Assignments	40%
Language Study Assignments	20%
Exams	30 %
Total	**100%**

5. 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

GENERAL INFORMATION

1. The GPA System used by the University Registrar's Office is:

A = 4.0	B = 3.0	C = 2.0	D = 1.0
A- = 3.66	B- = 2.66	C- = 1.66	D- = 0.66
B+ = 3.33	C+ = 2.33	D+ = 1.33	F = 0.0

2. Method of Instruction:

The following methods of instruction will be included in this course:

1. Lecture
2. Written Reports
3. Programming Assignments
4. Use of the Internet
5. Reading

3. Posting of Grades:

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

Tentative Schedule

Weeks 1~2	Language translation: Syntax and Semantics
Weeks 3~4	Inheritance and polymorphism
Week 5	Names, scoping, and binding
Week 6	Memory management
Week 7	Type checking
Week 8	Function calls
Week 9	Review and midterm
Weeks 10~12	Templates and generic programming
Weeks 13~14	Function programming and logic programming
Week 15	Review