BIOLA UNIVERSITY

Knowledge Representation and Automatic Reasoning

CSCI 480

SEMESTER (Spring 2017)

PROFESSOR/CLASS INFORMATION

Shieu-Hong Lin

(Course) Title: Topics in Computer Science
Theory of Computation
Term: Spring, 2017
Location: Busn 210
Office Phone: 562 903-4741
Office Hours: M-Th 8:30-10:30am,
E-Mail: shieu-hong.lin@biola.edu

Class Website: http://csci.biola.edu/csci440TC
Course Code/#: CSCI 440TC
Class Days/Time: T Th 1:30-2:45pm
Credit Hours/Units: 3
Office Location: Grove 8
Meetings with Professor: Make Appt via Email
Admin Assistant: Jerrianne Smith, x4741

COURSE ALIGNMENT WITH PROGRAM LEARNING OUTCOMES

CSCI 480 Research Seminar: Knowledge Representation and Automatic Reasoning. This upper-division course is an elective course for computer science juniors and seniors. Successful completion of this course (see next section) will prepare students to demonstrate a developing proficiency toward the accomplishment of PLO: analysis, modeling and problem solving.

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:

- write programs using Eclipse, a powerful programming language for constraint logic programming, and develop an in-depth understanding of the syntax and semantics of logic programming and the roles of constraint processing and search in problem solving,
- analyze constraint satisfaction problems, model the problem-solving schemes as Eclipse logic programs, and solve the problems by the search capability provided by Eclipse,

- understand logic and its important applications in automatic deduction, formal verification, and problem solving, and
- gain a solid understanding of computational complexity such as the concepts of complexity classes P, NP, NP-hard, and NP-complete and the implications of whether P=NP in computation.

REQUIRED TEXTS

- Krzysztof R. Apt and Mark, Constraint Logic Programming using Eclipse, Cambridge University Press, 2007.
- J. Kleinberg and E. Tardos. *Algorithm Design*, Addison Wesley, 2005.

LEARNING TASKS (Assignments) & ASSESSMENT (Grading)

Description and Weighting of Assignments:

Task 1: Weekly Reading and Progress Report

Due Date: Wednesday of the week **(15 assignments)**
Weighting: 15%
Possible Points: 3 points each.
Description: Using the template for cumulative weekly progress report, the student needs to incorporate information such as the amount of time he/she spent for the reading, attendance, and the overall progress in reading, programming, and other assignments since last Wednesday into the cumulative progress report.
Assessment: The student need to **(i)** finish the reading on time and record it in the progress report (1 point), **(ii)** attend the class this week (1 point), and **(iii)** gain a good understanding of 80% or more of the contents **or** have spent at least three hours in the reading (1 point).

Task 2: Weekly Programming Assignments (10 assignments)

Due Date: Wednesday of the week
Weighting: 45%
Possible Points: 6 points each.
Description: There will be 9 weekly 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.

- Peer discussion is most encouraged, but any copy-and-paste code from other people's programs is absolutely prohibited and will lead to serious discipline actions.
- Peer discussion based on the code shown on the screen and paper could be very helpful for debugging purpose and explanation of ideas. But you should never pass around your code as electronic files to others except for the TA and the instructor.
- You should make sure that you are able to reconstruct your code from scratch without any outside help when you submit a programming assignment as your own work.

Assessment: The student needs to email the source code files (all related .cpp and .h files) together with a self-evaluation report to the TA. 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: Exams (Quizzes and exams)

Weighting: 40%

Possible Points: Up to 50 points each.

Description: The exams have both the written component, which mainly tests the conceptual understanding of computation and logic, and the programming component, which tests skills in logic 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 details of logic programs using the computers in the lab. Missing the class may seriously hamper your understanding of many key concepts and programming skills critically needed in your programming assignments.

Policy Class attendance is counted toward points for the weekly progress report.

2. 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 6 more days after the due date and 1 point will be deducted from the report for a late submission before the submission is closed.

3. Turning in Assignments:

Assignments are expected to be received electronically submitted under Canvas.

4. Computation of Final Grade:

Weekly Progress Report	15 %
Weekly Programming Assignments	45%
Exams	40 %
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

Tentative Schedule

- Week 1 Logic programming and pure Prolog
- Week 2 A reconstruction of pure Prolog
- Week 3 Arithmetic in Prolog
- Week 4 Control and meta-programming
- Week 5 Manipulating structures
- Week 6 Constraint programming: a primer
- Week 7 Iteration in ECLiPSe
- Week 8 Top-down search with passive constraints
- Week 9 The suspend library
- Week 10 Constraint propagation in ECLiPSe
- Week 11 Top-down search with active constraints
- Week 12 Optimization with active constraints
- Week 13 Constraints on reals
- Week 14 Linear constraints over continuous and integer variables
- Week 15 Applications
- *Final*