CSE423: Compiler Writing, Spring 2022
Dr. Clinton Jeffery

Course Learning Outcomes (CLOs)
At the end of this course, a student should be able to:
1. Understand the theory behind compiler algorithms and be able to apply them in the context of solving compilation problems.
2. Understand each stage of compilation and be able to apply the appropriate transformations both manually and in code.
3. Apply the theory of compiler design and software engineering principles to build a working compiler.

Assessment Methodology:
***** Each CLO was tied to a measurable metric, either a question on the final, a student survey, or part of the main project assignment. These measurements did not overlap.
***** A formula was used to compute a normalized weighted sum from the scores for those questions and assessments. Raw scores were used in cases of direct questions.
***** The results were averaged over the entire class to compute a numeric score per outcome.
***** Only undergraduate (CS major) student data was used for this analysis.

The formulas used were:

CLO	Metric
1	Midterm exam problem #5 on a scale of 0-40
2	Final exam problems #7 and #12 on a (combined) scale of 0-50
3	Project grade on a scale of 0-110

For CLO 1, Midterm problem #5 was used. This problem required the students to analyze a context free grammar, identify its component parts, and explain differences between two different forms recursive grammar rules.

For CLO 2, Final problem #7 and #12 were used. These problems required the students to demonstrate understanding of issues in the area of symbol tables and intermediate code generation, respectively.

CLO3 was based on six scored phases/deliverables for the main project that was a large software engineering undertaking bringing to bear all of the theory presented in the course.

Performance Metric

This course, especially its programming project, is more difficult than most in the CS curriculum. The metric used to analyze outcomes for the course is:

Class Average	Performance Threshold
< 50%	Unsatisfactory
50% to <60%	Marginal
60% to <90%	Satisfactory
90% to 100%	Excellent

Results

Course Learning Outcomes

CLO	Class Average	Performance
1	32.77/40 = 82%	Satisfactory
2	38/50 = 76%	Satisfactory
3	61/110 = 55%	Marginal

Course Learning Outcomes 1-2 were satisfactory. Course Learning Outcome 3 was marginal.

Computation of Student Outcomes (SOs):

This course affects SOs 1, 2, and 6 (outcomes defined at the program level). We deal with each in turn by substituting a numeric value for performance (1,2,3, and 4 for unsatisfactory, marginal, satisfactory, and excellent respectively) and computing the average.

SO #1: Analyze a complex computing problem and apply principles of computing and other relevant disciplines to identify solutions. This outcome involves CLO 1.

CLO #	Performance	Overall
1	82%	Satisfactory

SO #2: Design, implement, and evaluate a computing-based solution to meet a given set of computing requirements in the contest of the program's discipline. This outcome involves CLO 3.

CLO #	Performance	Overall
3	55%	Marginal

SO #6: Apply computer science theory and software development fundamentals to produce computing-based solutions. This outcome involves CLO 2.

CLO #	Performance	Overall
2	76%	Satisfactory

Resulting Student Outcomes

SO	Title in Brief	CLO	Score	Interpretation
1	Analyze	1	3	Satisfactory
2	Design, implement, evaluate	3	2	Marginal
6	Apply theory	2	3	Satisfactory

Remedial Actions

Failure to Write a (toy) Compiler: In comparing project grades in Spring 2022 with the previous year, it is apparent that many students were unable to deliver working code in the later phases: semantic analysis and code generation. There was a strong bifurcation, with 22 students successfully delivering most of these latter phases, and 14 students receiving few or no points on these assignments. The course final grades were similar, with a high number (11/36) of students receiving a D or an F.

The reasons many students failed on their coding may be partly due to the instructor, the difficulty of the language that students were asked to write a compiler for (a Java subset), or the students being less prepared for a large-scale coding effort than in previous years. For example, in 2021 CSE 325 Operating Systems, which was supposed to deliver a medium-scale programming group project experience in preparation for CSE 423, reduced or eliminated group work and omitted the culminating final (filesystem) project due to covid.

Remediation via a Revised Group Compiler Project: In 2021 the CSE 423 project was individual, partly due to covid and also in order to preclude the likelihood of one student doing all the work on behalf of others on the team who do not contribute or learn the material as intended. In Spring 2022 I made groups optional, feeling like the number of students affected by covid or otherwise attending remotely would still impair their ability to do full group team projects. In Spring 2023 I will change the semester project to make it more appropriate for a group project and proceed with a software-engineering-style team orientation for most assignments. I will also change the language for which students are asked to write a compiler and provide more hands-on assistance during the scheduled lab hour and in required team meetings. Hopefully the group software-engineering style will improve students' ability to get their compiler working successfully.