CSCE [45]600 Syllabus

Spring, 2016

Instructor: Philip Sweany
Office: NTDP F262
Phone: 940-369-7427
Office Hours: Tuesday and Thursday from 9:00am to 11:30am; Wednesday from 9:00am to 10:50 am;
Email: sweany@cs.unt.edu

Textbook:

Teaching Assistant: Danielle Gaither (DanielleGaither@my.unt.edu)

Course Description:

CSCE [45]650 is a “first” compiler course. In it, students will build a new compiler for a language whose syntax and semantics is based upon a mixture of features from several modern imperative programming languages. This language will include a major subset of syntax and semantics found in imperative languages. In the process of building a compiler, each student will learn the basics of 1) using context-free grammars and regular expressions to define programming languages, 2) describing how parse tables for such grammars can be built, and 3) using such parse tables to parse strings that represent programs in such a programming language. As a by-product of all of this students will strengthen their overall programming skills and learn to use lex and yacc tools to build a parser for a given grammar.

Warning:
This course requires students to complete a significant program. Likely this program, writing a compiler, will be the hardest (if not the longest) programming project of your educational experience.

Course Outcomes:
Course outcomes are measurable achievements to be accomplished [by the student] by the completion of a course. Outcomes are evaluated in ABET’s accreditation process.
1. Given a context-free grammar, build SLR(1), LR(1) and LALR(1) parse tables.
2. Given a context-free grammar, an LR parse table and an input string, show the steps of the parse.
3. Given a language specification for an imperative language, build a parser for the language using tools such as lex and yacc.
4. Integrate semantic actions into the above parser to construct a symbol table, perform type checking, and generate intermediate code.
5. Given a control-flow graph with intermediate 3-address code within each basic block, show the “improved” control-flow graph after hand-optimizing for common subexpression elimination, copy propagation, and dead code removal.

Policies:

- Most programs are due at 11:59pm on the due date. Sometimes, if we use the class time as a lab session, program(s) may be due by the class period.
- No late homework will be graded.
- See the class web page for Program grading guidelines, both late policy and “re-bates.”
- All programs will be submitted to Blackboard.
- Except for the start of the term, attendance will not be taken in lecture. However, your attendance is strongly recommended to improve your opportunity to meet course outcomes.

Grading: I will use two different grading schemes, each of which will include one midterm, 2 homework assignments, one final and one program (with several deliverables.) Your grade will be based upon your average score, but I will compute two averages for each students and use the higher score to determine your final letter grade. I WILL guarantee that if your average is at least 90% you will get an A for the course; If your average is at least 80% you’ll get at least a B; if your average is at least 70 %, you’ll get at least a C; If your average is less than 70% you’ll get at least an opportunity to take the course again at some point in the future.

Grading Scheme 1

- Midterm exam 25% of your grade
- 2 homework assignments together weighted 25% of your grade
- One program (with several deliverables) weighted 30% of your grade
- Final Exam 20% of your grade

Grading Scheme 2

- Midterm exam 15% of your grade
- 2 homework assignments together weighted 15% of your grade
- One program (with several deliverables) weighted 60% of your grade
- Final Exam 10% of your grade
Make-Up Policy:

There will be no make-up exams, or programs given in this class. However, for documented *excused absences or emergencies* on a day of an exam the grade will be replaced by an average of the other exams. There is one exception to this rule. Under NO circumstances will more than one exam be replaced by an average of the other scores. For a second (and subsequent) missed exam or lab, even if all are excused, students will receive a 0 for the missed work.

**Excused Absences**

Students are expected to schedule routine appointments and activities to not conflict with attending class. However, some absences cannot be prevented. In the event of a medical *emergency* or *family* death, students must request an excused absence as quickly as feasible following the emergency. Use common sense. Students must provide documentation to verify the emergency.

**Emergencies**

By definition, emergencies cannot be planned for. Your instructor attempts to make accommodations in these instances that allow for making up missed work and completion of the course in a timely manner. Among these emergencies are

- A death in your *immediate* family
- An accident or illness requiring immediate medical treatment and where a doctor has indicated attending class is impossible or inadvisable.
- Employees on call 24/7 must document that they were called during class time.

Attendance Policy:

Except for the start of the term, when I'm required to take attendance to obtain an accurate count of enrollees, I will not generally be taking attendance.

Collaboration and Cheating:

Do **NOT** work with other students on shared program solutions. Do **NOT** get help with algorithms or coding from anyone other than Dr. Sweany or the TA. Do **NOT** use even partial program solutions from the internet unless those partial solutions are provided to you by Dr. Sweany as part of the assignment description. Failure to adhere to these strict standards will be cause for disciplinary action that could be as severe as expulsion from the university.

It **IS** permissible to obtain help from whoever you wish to fix *syntax errors*. But remember, for anything but syntax errors, getting programming assistance from any source other than Dr. Sweany or this class’ TA will be considered cheating and dealt with harshly.

And, of course you need to do your own work on exams as well. Here there should be no ambiguity at all.

In case the above description, and in-class discussion of my views on appropriate and inappropriate collaboration does not answer all of your questions, please look at the university
Pair Programming:

There is **ONE** exception to the section above when it comes to the compiler program that you’ll be working on this term. You MAY work in pairs on the compiler. **NOTE:** You do NOT need to work as part of a pair. That is merely your choice. I’ll explain the option(s) and reason(s) for them in class, but IFF you choose to work as a pair your joint compiler will require additional feature(s). Again this will be discussed in class.

SPOT

The Student of Teaching (SPOT) survey is a requirement for all organized undergraduate classes at UNT. This short survey will be made available to you at the end of the semester, providing you a chance to comment on how this class is taught. I am very interested in the feedback I get from students, as I work to continually improve my teaching. I consider the SPOT to be an important part of your participation in this class.

ADA

UNT complies with all federal and state laws and regulations regarding discrimination, including the Americans with Disability Act of 1990 (ADA). If you have a disability and need a reasonable accommodation for equal access to education or services please contact the Office of Disability Accommodation.
### Tentative Schedule

<table>
<thead>
<tr>
<th>Topic</th>
<th>Chapter</th>
<th>Class Period</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>1-2</td>
<td>1-2</td>
</tr>
<tr>
<td>Lexical Analysis</td>
<td>3</td>
<td>3-4</td>
</tr>
<tr>
<td>Syntactic Analysis</td>
<td>4</td>
<td>4-11</td>
</tr>
<tr>
<td>Syntax-Directed Translation</td>
<td>5</td>
<td>12-15</td>
</tr>
<tr>
<td>Code Generation I</td>
<td>6</td>
<td>16-19</td>
</tr>
<tr>
<td>Run-Time Environments</td>
<td>7</td>
<td>20-22</td>
</tr>
<tr>
<td>Code Generation II</td>
<td>8</td>
<td>23-25</td>
</tr>
<tr>
<td>Machine Independent Optimizations</td>
<td>9</td>
<td>26-27</td>
</tr>
<tr>
<td>Final Exam</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>