CpS 450 Course Syllabus

College of Arts and Science

Instructor: Sarah Gothard, Ph.D.
Email: sgothard@bju.edu
Office: Alumni 84

Telephone: (864) 242-5100 ext. 8152
(937) 321-5167 for urgent texts

Office Hours: MWThF 8:15-9:45 am
MWF 12:00-12:45 pm *by appointment only*
T 8:15-10:30 am
TTh 2:00-2:45 pm

Required Material

Compiler Design: Theory, Tools, and Examples, by Seth Bergmann. Available Online

Course Description

A study of compilers and interpreters, including scanning, parsing, and code generation. A compiler will be implemented with the aid of compiler generation tools. Second semester, three hours.
Prerequisite: CpS 350.

Course Overview

This course provides an introduction to the issues involved in translating high-level computer languages to low-level ones. Compiler-generation tools are the result of one of the most successful examples of the application of theory to practice in the history of computer science, and their discussion will receive special emphasis.
As a requirement for passing this course, the student must successfully implement a compiler that meets certain minimum specifications. The compiler project is divided into several deliverables and checkpoints. A successful project will be the result of timely submission of deliverables as well as a working final product.
Students who fail to meet the minimum project specifications will not pass the class.

Course Context

This course supports the following objectives of the Computer Science and Information Technologies programs:
CS 2. Use appropriate technology as a tool to solve problems in various domains
CS 5. Demonstrate an ability to communicate technological information effectively both in written and oral forms
CS 6. Demonstrate an ability to acquire new knowledge in the computing discipline

Learning Objectives

Objective	Content	Assessment

Describe the steps and algorithms used by language translators (CS 8)	(Most lectures)	Test 1, 2, Final
Use compiler generation tools to assist in the construction of a compiler (CS 2)	Lecture 1, 3, 8	Project
Recognize the connection of formal models such as finite state automata to language definition through regular expressions (CS 8)	Lecture 2	Test 1
Discuss the types and effectiveness of optimization (CS 8)	Optimization Lecture	Final

Grading

Qty	Item	Points	Total	**Scale:**	
3	Exercises	Varies	100	A	90-100%
4	Deliverables	Varies	500	B	80-89%
2	Written Tests	Varies	280	C	70-79%
1	Final Exam	120	120	D	60-69%
Total Points:		1,000		F	<60%

Course Policies

Work is due at the deadline. **Late work receives a 0**. Extensions may be purchased with tokens.

Do not share class notes with anyone who is not enrolled in the same class section as you are during the same semester.

You may not use generative AI tools (i.e. Chat GPT, Bing Chat, Google Bard, etc.) in this course for any assignment without the professor's express permission. Should an AI tool be used with permission, its use must be documented.

Compliance with student handbook policies is expected during class.

Professionalism

University classes are a place to sharpen your professional habits. Arrive on time. Dress appropriately. Stay alert. Engage with the material. Take pride in your work. Build relationships. Encourage growth in others.

University Policies

Handbook Policies

Compliance with student handbook policies is expected during class.

Attendance Policy

You are expected to attend class and be on time: https://home.bju.edu/bju-policies/. A partial attendance will be recorded when you miss the beginning or end of a class. If you miss more than 15 minutes of class, you will be marked absent. Students who exceed the allowed absences may be withdrawn from class.

If you need to miss class any reason, please contact me as soon as possible. Assignments and tests should be completed before planned absences.

Accommodations for Students with Disabilities

Students are required under Section 504 to communicate the need for accommodations and provide documentation to the Academic Resource Center Accommodations Office in AL 213.
Visit https://success.bju.edu/ for more information. Students are encouraged to seek an appointment in the first week, as accommodations are not provided retroactively.

Academic Honesty and Integrity Policy

See the Computer Science Department's Academic Integrity Policy:

https://cs.bju.edu/academics/policies/academic-integrity-policy/

Taking credit for someone else's work is unethical in any setting. In a university setting, it undermines the ability of faculty to accurately evaluate your competence, harming you and the reputation of the department. For these reasons, the penalties for academic dishonesty may be severe.

Cheating on assignments and tests is forbidden. All work is to be done individually unless group work is explicitly permitted. No collaboration is allowed on tests. For regular individual assignments, we expect that the submitted work represents the student's own intellectual effort, defined as follows:

1. The program was written primarily by the student. This means that most of the code (aside from starting code provided by the instructor) must have been crafted, not copied, by the student.
2. External resources used, whether electronic or from another human, must be documented as follows:
    - Code snippets copied from online resources must be documented by a comment just above the copied snippet giving the URL of the page containing the source.
    - Explanatory help or advice regarding the design or implementation of the solution received from people other than the instructor must be documented in a report accompanying the assignment submission. This report must detail:
        - Source of information (e.g., name/email of the person who helped)
        - Relevance (i.e., how this resource helped and/or what it provided)
    - Note that students must not consult a solution to the assignment as a resource in crafting their own solution, nor share their own solution with another student. Doing so constitutes cheating.
3. The student must be able to explain, on demand, the entirety of the program on both the syntactic and semantic level.

Failure to comply with any relevant integrity requirement constitutes cheating. Such incidents will be reported to the academic integrity committee. To avoid trouble:

- Do not look at another student's program code when seeking assistance. On the other hand, if another student is seeking help from you, never use your own program code as an example. The only acceptable reason another student may look at your code is to help you find a problem in your program.
- Do not write program code while another student (or lab assistant) is sitting with you. You may work out designs in pseudocode on paper with another student, but you must write program code by yourself.
- When seeking assistance from another person on a program assignment, always get his/her name so you can fulfill the documentation requirements.

Generative AI

Since the goal of the assignments in this course is to learn to develop the skills covered NOT complete the tasks assigned, and since the use of AI to complete or jump start tasks defeats the goal of the assignments, you may not use generative AI tools (e.g., GitHub Copilot, Amazon CodeWhisperer, Tabnine, CodeWP, OpenAI Codex, CodeT5, Chat GPT, Bing Chat, Google Gemini, etc.) in this course for any assignment without the professors express permission. Should an AI tool be used with permission, its use must be documented.

Tentative Schedule

Date	**Class**	**Reading**	**Assignment**
Thu, Jan 16	Introduction	CD, Ch. 1	
Tue, Jan 21	Scanning	CD, Ch. 2	
Thu, Jan 23	Scanning		
Tue, Jan 28	Syntactic Analysis	**CD, Ch. 3**	**Phase 1**
Thu, Jan 30	Recursive Descent Parsing ANTLR Parser Generation	**CD, Ch. 4-p.104**	**Ch. 3 Homework**
Tue, Feb 04	Grammar Analysis	**CD Ch. 4 pp. 105-121**	**Ch. 2 Homework**
Thu, Feb 06	LL Parsing Test 1 Review		**Phase 2 Checkpoint**
Tue, Feb 11	ALL Parsing		**Ch. 4 Homework**
Thu, Feb 13	**Test 1**		
T-F, Feb 18-21	*Bible Conference*		**Phase 2**
Tue, Feb 25	ANTLR Tree Processing, Symbol Table		
Thu, Feb 27	Semantic Processing		

Tue, Mar 04	Code Generation		
Thu, Mar 06	Code Gen – Variables		**Phase 3**
Tue, Mar 11	Code Gen – Control Structures		
Thu, Mar 13	Code Gen – Method Calls		
Tue, Mar 18	Code Gen – Debugging		
Thu, Mar 20	**Test 2**		
M-F, Mar 24-28	*Spring Break*		
Tue, Apr 01	Code Gen – Runtime Storage		**Phase 4**
Thu, Apr 03	Multiple Classes		
Tue, Apr 08	Strings, Semantics – Inheritance		
Thu, Apr 10	Code Gen - Inheritance		
Tue, Apr 15	Garbage Collection		
Thu, Apr 17	Optimization	CD Ch. 7	
Tue, Apr 22	Optimization		
Thu, Apr 24	TBA		**Phase 5**
Tue, Apr 29	Peer Reviews		**Manual Draft**
Thu, May 01	Beyond Compilers Final Review		**Manual**
Mon, May 05 9:30–10:40 a.m.	**Final Exam**		

Copyright Policy