Theory of Programming Languages

Fall 2025

Instructor:	Dr. Deng, Qixin	**Time:**	Tues/Thur 2:40 PM - 3:55 PM
Email:	dengq@wabash.edu	**Place:**	Goodrich Hall, 101
Office:	Goodrich Hall 108	**Office Hour:**	Mon/Wed 10:00 am to 5:00 pm

Office Hours: Find me any time from 10:00 am to 5:00 pm on Monday/Wednesday. I will not be available on Tuesday and Thursday. A pre-schedule/appointment is required after 5:00 pm.

Course Description: Programming language theory (PLT) is a branch of computer science that deals with the design, implementation, analysis, characterization, and classification of formal languages known as programming languages. Functional programming languages, as ancestors of modern programming languages, many of their features have been inherited and are well developed today. They have much simpler and more straightforward structures and syntax and are much more user-friendly than Machine Languages. We will think from the creator's point of view, go deep into the principles of computer language design, and explore the initial design requirements of computer languages.

Reference Note:

1. Racket Official Documentation: https://docs.racket-lang.org/
2. Haskell Official Documentation: https://www.haskell.org/documentation/

Platform:

1. Racket: https://racket-lang.org/download/
2. Haskell: https://www.haskell.org/downloads/

Course Goal:

- **Understand programming paradigms and formal methods**: Compare functional and imperative programming, and apply formal methods such as syntax, grammar, parse trees, abstract syntax trees, and lambda calculus to analyze and construct programs.
- **Design and implement functional solutions**: Write recursive, higher-order, and type-safe programs using functional languages, while exploring advanced ideas such as streams, backtracking, and macros.
- **Master type system fundamentals**: Analyze strong vs. weak typing, static vs. dynamic typing, algebraic data types, and polymorphism to reason about program correctness and structure in Haskell and other languages.

Academic Credit Statement: We will use two new programming languages in this course. Unlike CSC-111 where students learn Python programming, in this course, students will use Racket and Haskell as platforms to explore the basic design of the theory of programming languages. To support students' learning beyond the classroom, I will provide structured virtual lab sessions where students can gradually build skills in both languages under guided instruction.

- Students will participate in 7 virtual lab sessions for Racket, each of which includes notes, demonstrations, and hands-on practice with several Racket programming exercises. Each session is designed to take roughly 1 hour and will account for **7** *additional 4th hour sessions* for the semester.

- Students will participate in 7 virtual lab sessions for Haskell, each of which includes notes, demonstrations, and hands-on practice with several Haskell programming exercises. Each session is designed to take roughly 1 hour and will account for **7** *additional 4th hour sessions* for the semester.

In total, the course includes 14 hours of additional structured virtual lab activities outside of regular class meetings (equivalent to 1 hour per week for 14 weeks.)

Prerequisites: CSC-111

Grading:

- 50% Assignments
- 50% Exams
- I will apply the letter grade criteria as below:

Letter	A	A-	B+	B	B-	C+	C	C-	D+	D	D-
Total	>=93	>=90	>=87	>=83	>=80	>=77	>=73	>=70	>=67	>=63	>=60

Class rules: The student who shows discrimination, disrespect, bullying, and other bad behaviors during the lecture will immediately fail the course. And I will report to both the department and the college.

Assignments: There will be 10 assignments across the semester. Each assignment will have several programming questions. I do not accept late submissions. Please submit the assignment on time.

Exam:

Midterm Exam .....Oct. 14th, Tuesday, in class time
Final Exam .....Dec. 16th, Tuesday, 1:30 pm - 3:30 pm.

Guidelines for Ethical Computing and Working with Others
For the purposes of this course, adhering to the Gentleman's Rule means also adhering to the ACM Code of Ethics and Professional Conduct. The Code lays out the fundamental ethical principles that should guide all students and professionals in their study and work when computing and technology are involved. One major part of ethical computing is respecting the ideas and works of others, and only using them in an ethically appropriate and responsible manner. Computing, like almost all human endeavors, is a collaborative enterprise. We learn from each other and from the work of those who came before us, just as our descendants will learn from us. To not respect the works of others damages one's own learning, and ultimately damages human society as a whole.

For the purpose of this course, I will add the following guidelines. In general, unless specifically authorized for a particular assignment, you should refrain from providing program code or solutions to another student, or receiving program code or solutions from another student. More concretely: providing code or solutions to another student or receiving code or solutions from another student, unless specifically authorized, constitutes academic dishonesty and a violation of the Gentleman's rule. This applies also to both past and future students in this course.

For the purpose of this course, I am happy for students to work in collaborative partnerships. You can discuss it together, but you have to finish your exercises by yourself! Collaborative means learn from each other, do not copy-paste! You should not turn in anything unless you fully and completely understand it and can fully and completely explain it.

In order to foster your ability to communicate about technical matters, any assignment may include an interview with me, during which I will ask you to explain your code or solution and the process by which you came to it. While I acknowledge that technical communication is a skill that must be practiced to be improved, exceptionally poor performance during an interview will raise my suspicions that you may be in violation of the Gentleman's Rule and the ACM Code.

Academic Credit Policy
This course complies with the Wabash College Academic Credit Policy stated in the Academic Bulletin. In addition to regular class meetings, direct instruction takes place in this course through: faculty office hours, advising for group and individual projects, technical interviews, through Canvas, online videos, and public facing activities.

The Syllabus as a Living Document
A syllabus is a living document and can be amended. I will communicate any syllabus changes; it is your responsibility to be aware of course policies at all times.

Quantitative Skills Center (QSC)
Is biology shredding your brain into fine particulate matter? Does chemistry have your anxiety level reacting negatively? Do you have bugs in your computer science code that you just can't squash? Are the demands of your economics homework far outpacing your supply? Is math leaving you feeling irrational or derivative? Has your last physics exam, not gravity, been pulling you down? If you have questions about biology, chemistry, computer science, economics, mathematics, or physics, visit the Quantitative Skills Center (QSC)! Our department-selected tutors will work with you to answer your questions and deepen your understanding of each subject. The QSC operates Sundays, Tuesdays, and Thursdays from 7- 11pm on the second floor of the Library. Please check out the QSC webpage `https://www.wabash.edu/ace/qsc` for specific tutor availability. If you have any questions, please email Vic Lindsay at lindsayv@wabash.edu.

Writing Center
Do you have questions about how to start a paper? Are you struggling to get all of your ideas to fit? Do you have a draft but want someone to review it? Are you unsure of how to incorporate comments from your professor? Head to the Writing Center! No matter what your writing questions or needs, the Wabash Writing Center. Consultants are eager and able to help you! Please visit the webpage, `https://www.wabash.edu/ace/writing` to schedule an appointment. If you have any questions, please email the director, Dr. Koppelmann, at koppelmz@wabash.edu.

The Office of Student Enrichment
Succeeding at Wabash College takes a great deal of effort and planning. Life is complex, assignments are time consuming, and staying involved keeps you running. When you have questions about how to make everything fit into your schedule, how to study more efficiently, how to take better notes, or any other question about developing your college skills, visit the Office of Student Enrichment (OSE). Go to `https://www.wabash.edu/ace/office` and follow the "Make an Appointment" link to arrange a one-on-one, personalized meeting with Dr. Koppelmann or one of the OSE Advisors. No matter your questions, we will work with you to find a solution that helps you achieve your goals.

Disability and/or Special Accommodations
Students with disabilities (apparent or invisible) are invited to confidentially discuss their situation with the disability coordinator, Heather Thrush, Associate Dean for Student Engagement and Success. If a student wishes to receive an academic accommodation, it is required that his documentation of the disability be on file with Dean Thrush, who can, in confidence, provide information and guidance. Early notification helps us all work together in the most effective ways. To make an appointment with Dean Thrush, call (x6347), or email thrushh@wabash.edu. Appointments can be available via Zoom, if needed.

This is a tentative schedule, I will adjust it a little bit based on the feedback. Considering there will be holidays and exams during the semester, I only prepare and list 13 weeks of course content arrangement.

Course Content Arrangement	
Week #	Course Content
Week 01	Learning basics of Racket and Haskell
Week 02	Introduction; Imperative Programming VS Functional Programmming; Syntax and Grammar; Semantics; Lambda Calculus
Week 03	Computation on Recursive Datatypes; Patterns for Pattern Matching; Single Linked List; Tail-call Optimisation
Week 04	Higher-order Functions on Lists: Map, Filter, Fold,...; Currying and Partial Evaluation; Functions Returning Functions
Week 05	Static and Dynamic Scope; Strict and Non-Strict Evaluation
Week 06	Object Systems; Pattern-Based Macros
Week 07	Self-Reference in Classes; Generators; Lazy Evaluation; Streams
Week 08	The Ambiguous Operator and Non-Deterministic Expressions; Local Mutation; Continuations
Week 09	Continuations (continued); Backtracking; Type Systems
Week 10	Type Systems; Parametric Polymorphism
Week 11	Typeclasses; Ad-Hoc Polymorphism
Week 12	Functors and Monads; Option, Either, and List as Monads
Week 13	The State Monad; The IO Monad