Programming Languages
CS 331 - Spring 2019
Credits: 3 hours

Instructor: David Furcy **Office:** Halsey 221
Email: furcyd@uwosh.edu **Phone:** 424-1182
Office Hours: MWF 10:10AM-11:10AM, TuTh 9:30AM-11:00AM, or by appointment
Class Meetings: MWF 12:40PM-1:40PM in HS 202
Prerequisites: A grade of C or better in CS 271
Class Web Page: D2L
References:

- A draft/in-progress interactive online book I am currently working on with Dr. Naps hosted at *canvas.instructure.com* : use the following link to the CS331 - Spring 2019 course: https://canvas.instructure.com/enroll/C8AANC **You MUST use your *uwosh.edu* email address** to set up your account at Canvas so that I can give you appropriate credit for doing the interactive, practice problems that are interspersed throughout the book.
- Additional materials (e.g., slides and other handouts or links) will be posted on D2L no later than the day before the class meeting where they will be discussed.

If you have special needs, please come and talk to me as soon as possible so I can accommodate your needs right away.

Students are advised to see the following URL for disclosures about essential consumer protection items required by the Students Right to Know Act of 1990:

https://uwosh.edu/financialaid/consumer-information/

Learning Outcomes:

- Given an English description of a formal language, the student will be able to construct a context-free grammar that generates this language.
- Given a context-free grammar and the source code for a program, the student will be able to parse the program according to the grammar, and to produce a parse tree of the program or identify syntactic errors in the program.
- Given a context-free grammar in Backus-Naur Form (BNF), the student will be able to convert it to an equivalent, more compact grammar in Extended BNF.
- Given a context-free grammar, the student will be able to determine whether the grammar is ambiguous or not.
- Given a program and a scoping mechanism (static or dynamic), the student will be able to trace the execution and infer the output of the program.
- Given a formal description of an operation, the student will be able to implement it in the functional paradigm using either recursion or a computational pattern such as filtering, mapping, or folding, or a combination thereof.
- Given an imperative program and a set of eager/lazy parameter-passing mechanisms, the student will be able to simulate, for each mechanism, the sequence of updates that take place in memory as the program executes.

- Given the description of an operation applicable to an infinite data structure, the student will be able to program this operation in a functional language using lazy evaluation.
- Given a functional language with higher-order functions, the student will be able to simulate recursion using the Y combinator.
- Given a working interpreter for a programming language, the student will be able to adapt the interpreter to a similar language with a different concrete syntax.
- Given a working interpreter for an imperative programming language, the student will be able to implement an interpreter for an enhanced language with additional features (such as a new data type or a new language construct), or different semantics (such as a different parameter-passing mechanism).
- Given a problem involving different variants of a data type (e.g., different types of expressions) and different operations on those data (e.g., evaluation, pretty-printing, type-checking, etc.), the student will be able to compare and contrast the procedural/functional approach and the object-oriented approach and to discuss the strengths and weaknesses of each approach. Given a desirable property of a program or a software development process (e.g., programming convenience, ease of prototyping, support for code reuse and code evolution, high runtime performance, etc.), the student will be able to discuss the pros and cons of static checking and dynamic checking with respect to this desirable property.
- Given a programming language and its type system, the student will be able to define the properties of soundness (or safety) and completeness of this type system, and to compare and contrast weak typing and strong typing.

Topic Coverage: We will cover the following topics:

- Grammars and formal syntax
- The functional programming paradigm
- Recursion in programming
- Scope and lexical structure of programs
- The lambda calculus as a formal model of functional languages
- Writing interpreters for small languages
- Evaluation order and parameter-passing
- Type systems

Note that the catalog course description shown below is out of date:

> *A study of programming languages. Topics covered include: formal syntactic description, methods of implementation, and language features such as recursion, block structure, string processing, and list processing. Specific high level programming languages are studied to demonstrate the use of these language features.*

Course Grading Policy

Your final grade for this course will be based on three components, namely exams, assignments, and class preparation/participation. Your overall numerical grade for the course will be computed as the weighted sum of the component grades using the weights in the following table:

Component	Weight
Exams (3)	55% broken down as: best exam score: 24% worst exam score:12% third exam score: 19%
Assignments (6-8)	35%
Class preparation/participation (daily)	10%

The final exam will take place during week 14. The first two exams will take place around weeks 6 and 10, respectively. Exam dates will be announced in class and on D2L at least one week before the exam.

Each assignment will be worth between 20 and 100 points. The corresponding 35% of your overall grade will be obtained by adding up all of your assignment scores and dividing the resulting total by the sum total of all assignments' worth. Exams do not carry the same weight either. Note that your best exam score counts twice as much as your worst exam score. The grading for class preparation/participation is explained below.

Finally, your letter grade for the course will be computed as follows:

Numerical Score	Grade	Numerical Score	Grade
≥ 92	A	≥ 72	C
≥ 90	A-	≥ 70	C-
≥ 88	B+	≥ 68	D+
≥ 82	B	≥ 62	D
≥ 80	B-	≥ 60	D-
≥ 78	C+	< 60	F

While this overall grading scheme is fixed, I will be happy to discuss any issue you may have with individual grades. If you notice a mistake or have a question regarding a specific grade, please come and talk to me *as soon as possible*. Do not wait until the end of the semester to bring up grading issues. Also, I will *not* be available to discuss grades after the end of the final week.

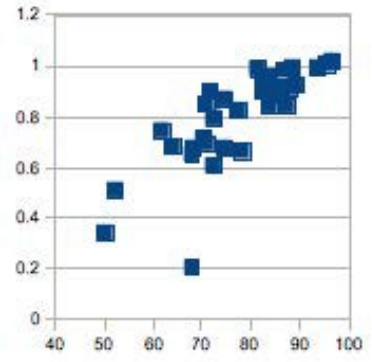
Attendance and Participation

You are expected to not only attend **every** class meeting but also to come **prepared** for and **participate** actively in it. Necessary preparation requires you to have studied and assimilated the material covered in previous sessions, to have completed the reading assignment for the day, to have met with me outside of class to discuss any questions you may have, and to have completed the assignments on time. One additional and crucial component of class preparation/participation are the daily practice problems. At the end of most of our class periods, I will make available to you a set of practice problems covering what we discussed in class that day. Research on effective studying skills and college success shows that the time to work on these practice problems is immediately after the material is covered in class. You are also encouraged to discuss practice problems with your classmates in a spirit of mutual help toward better understanding of how to solve them. We will discuss the practice problems at the beginning of the class following their distribution. In fact, at the beginning of each class, I will call on students

at random to discuss their answer to a specific practice problem. Each time the student's answer demonstrates that he or she has not worked hard enough on the problem (my call!), he or she will be penalized by one point for that practice problem. Your solutions to the practice problems are due no later than 10:00AM on the Monday, Wednesday, or Friday following the day of their distribution. These solutions must be submitted within the Canvas book web site, that is, by clicking the answer button (typically called "Check Answer") associated with each practice problem.

Each problem (i.e., practice problem or proficiency exercise) will be worth one point. The component of your overall course grade labeled "Class preparation/ participation" and weighted 10% will be obtained as the ratio of the sum of points earned over all practice problems to the total number of practice problems. If you have participated in class the day the practice problem was distributed, have made a good faith effort to work on the practice problem, and are stuck on it, I will be more than happy to help you with it if you come to my office hours or make an appointment within three days of the distribution of the practice problem. After those three days (not counting weekends), because you have made the choice *not* to learn effectively, you are on your own in terms of grappling with this particular practice problem.

Although the practice problems only count for 10% of your grade, the following correlation from a previous course between the total score on practice problems (on a 0 to 1.0 vertical scale) and the final course grade (on a horizontal scale of 0 to 100) is indicative of their true importance:



It is hard to imagine how a student could do well in this course while skipping (or not working conscientiously on) the practice problems, missing classes, or attending them unprepared (e.g., by not completing the daily reading assignments BEFORE class). On the positive side, I have high expectations for my students and will always support and encourage you. I **strongly encourage you to ask any question** or raise any issue you have with the course either during or at the end of class, or during office hours. I will also gladly meet with you by appointment (just send me an email with a list of time slots when you are available). While I will meet with you as soon as my schedule permits, do not expect me to be widely available just before an assignment is due or an exam is scheduled when . Hint: Start early and plan ahead!

Late Submissions

I will describe the submission procedure for your assignments when the time comes. However, let me point out right away that each assignment will come with a deadline (day and time,

typically 10:00AM) after which any submission will be considered late. The late-submission policy works as follows:

Turned in	Penalty
On the due date but after the deadline	20%
The day after the due date	30%
Two days after the due date	60%
More than two days after the due date	100%

Note that assignments more than two days late receive no points. **Weekend days and holidays count as "regular days" when computing late penalties**. Each (late) day starts precisely at midnight. So, each one of the following timestamps: 12:00:00AM, 12:00:01AM, etc., is considered to be "the next day." Extensions on assignments may be granted at my discretion if you provide a valid justification (in the form of a written excuse from a medical doctor or the Dean of Students Office) **before** the due date. Late submissions can easily be avoided by starting to work on the assignment right away and asking for help early if you get stuck.

NO late submissions will be accepted for daily practice problems. They *cannot* be made-up.

If you miss a scheduled exam, you **may** be able to take a make-up exam provided you give me a valid justification (see above), ahead of time if possible. Only one make-up exam will be given. It will be a comprehensive exam scheduled at the end of the semester.

Collaborating versus Cheating

You are encouraged to work with others when working on the practice problems, as long as everyone comes out of the collaborative work understanding and being able to apply to similar problems the knowledge or skills that the practice problem was targeting.

For assignments, you have the choice on each assignment to work on your own or with one partner, in effect working in "pair learning/programming" style. You are not allowed to collaborate with anyone else for assignments. If you decide to work with a partner, here is the way that it must be done:

- Both you and your partner must "declare" via emails sent to me, that you will be working together on an assignment. This declaration must be made **AT LEAST FIVE DAYS** prior to the deadline for the assignment on which you are working together.
- For that assignment, only one of you will submit the assignment. That is the assignment that will be graded. You will both receive the same grade.
- You may not share or even discuss your work with anybody but your partner unless you can live with a zero and the other potential academic sanctions of cheating (see the UWO Student Discipline Code, Chapter UWS 14).

If you decide to work on your own, then the third bullet point above also applies (minus the partner). Either way, you are of course encouraged to ask me for help if you get stuck after making a good faith effort at solving a problem. That is why I have regular office hours!

In conclusion, remember that computer science classes require a lot of work in addition to active participation in class. It takes considerable practice to develop the technical and analytical skills

targeted by this course. You will need to spend **at least (and typically more than) three hours of effort outside of class for each in-class hour**. Having said this, I expect every hardworking student to do well in this course.

Have fun this semester and good luck!