Instructor: Zena M. Ariola (ariola@uoregon.edu).

Piazza: We will be using Piazza for class discussion. The system is highly catered to getting you help fast and efficiently from classmates, GEs, and myself. Rather than emailing questions to the teaching staff, I encourage you to post your questions on Piazza. Find our class page at:

piazza.com/uoregon/spring2022/cis425

Office hours:

<table>
<thead>
<tr>
<th>Day</th>
<th>Time</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tuesday</td>
<td>10:00am-10:50am</td>
<td>Zena</td>
</tr>
<tr>
<td>Wednesday</td>
<td>10:00am-10:50am</td>
<td>Bosco</td>
</tr>
<tr>
<td>Thursday</td>
<td>10:00am-10:50am</td>
<td>Zena</td>
</tr>
<tr>
<td>Friday</td>
<td>10:00am-10:50pm</td>
<td>Bosco</td>
</tr>
</tbody>
</table>

If needed, the instructor can provide one on one help in a break-out room on a first come first serve basis.

Grading policy:

<table>
<thead>
<tr>
<th></th>
<th>%</th>
<th>Tentative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Midterm 1</td>
<td>30%</td>
<td>April 21 or April 26</td>
</tr>
<tr>
<td>Midterm 2</td>
<td>30%</td>
<td>May 17 or May 19</td>
</tr>
<tr>
<td>Final</td>
<td>40%</td>
<td>June 8</td>
</tr>
</tbody>
</table>

The time of the midterms might change depending on how much material is covered.

We will post weekly assignments, which however will not be graded and therefore you will not receive any points. Solutions to the assignments will not be posted online, but will be presented during the discussion session on Friday. Even though there are no points for the homework, we strongly urge you to do the assignment and then compare your solution with the one presented during the discussion session. In case your solution differs and you are not sure of the correctness of your solution, please send us your solution together with an explanation of why you think your solution is correct.
Grading rubric:

A **Excellent.** Solid grasp of the concepts and approaches introduced in the course
B **Very good.** Good grasp of the concepts and approaches introduced in the course
C **Acceptable.** Sufficient grasp of the concepts and approaches introduced in the course
D **No Pass (Earn UO credit).** Insufficient grasp of the concepts and approaches introduced in the course
F **No Pass (No credit).** Little or no demonstrated grasp of the concepts and approaches introduced in the course

Grading scale:

<table>
<thead>
<tr>
<th>Percentage Range</th>
<th>Grade</th>
<th>Grade</th>
<th>Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>90-100%</td>
<td>A</td>
<td>A+</td>
<td>98.67-100.00</td>
</tr>
<tr>
<td>80-89.99%</td>
<td>B</td>
<td>B+</td>
<td>86.67-89.99</td>
</tr>
<tr>
<td>70-79.99%</td>
<td>C</td>
<td>C+</td>
<td>76.67-79.99</td>
</tr>
<tr>
<td>60-69.99%</td>
<td>D</td>
<td>D+</td>
<td>66.67-69.99</td>
</tr>
<tr>
<td>0-59.99%</td>
<td>F</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Text:** We will cover material from the book *Concepts in Programming Languages* by Prof. John Mitchell. The book is available at the library: [https://www-cambridge-org.libproxy.uoregon.edu/core/books/concepts-in-programming-languages/1C05841AB47B49D12C7FC48D1022F11E](https://www-cambridge-org.libproxy.uoregon.edu/core/books/concepts-in-programming-languages/1C05841AB47B49D12C7FC48D1022F11E)

We will put pointers to the relevant chapters from the syllabus page on Canvas. On the syllabus page you will find pointers to the material discussed each week.

**Course Objectives:** The main goal of the course is to introduce you to the different aspects of programming languages, including their features, type systems, programming style, and implementation. We will consider several programming languages: ML, Haskell, Racket and Prolog. You will not become an expert programmer in the languages taught in this class. However, you will gain the necessary skills to understand the impact of language features on both the implementation and the abstractions they provide. We will cover different programming paradigms: imperative, functional, object-oriented and logic programming. Each paradigm provides a different way to solve a problem. **Note that this is not a survey course!**

**Learning Outcomes:** On completion of the course, students

- will be better programmers, even in languages not covered
- will be able to distinguish surface differences from deeper principles
- will be able to write programs in ML, Racket, Haskell and Prolog

Moreover, they must demonstrate a thorough understanding of the following topics:

- Compilation vs Interpretation: advantages and disadvantages in terms of speed of execution and memory space.
- Syntax vs Semantics: what are the legal expressions and what do expressions mean.

**Concepts:**
- Parse trees and Abstract syntax trees
– Lexical analyzer
– Parser
– Operational Semantics

• Imperative programming (Cobol, Fortran, C, Pascal, Ada, Ruby, Python): This is the closest model to the machine architecture.
  Concepts:
  – Side-effects
  – Types (static and dynamic, weak and strong)
  – Scope (static and dynamic)
  – Parameter passing techniques (call-by-value, call-by-reference, call-by-value-result, call-by-name)
  – Aliasing
  – Block-structured languages
  – Run-time structures (Stacks vs heap)
  – Garbage collection

• Functional programming (Racket, ML, Haskell): Based on mathematics which gives a theoretical foundation.
  Concepts:
  – Strict vs non-strict functions
  – Higher-order functions
  – Currying
  – Lazy vs eager evaluation
  – Innermost vs outermost evaluation
  – Type inference
  – Type checking
  – Overloading
  – Coercion
  – Polymorphism
  – Type classes
  – Pattern matching
  – Monadic approach to effects
  – Programming with infinite structures

• Deductive programming (Prolog): Based on the concept of a relation.
  Concepts:
  – facts and rules
  – search
  – unification - logic variable

• Control mechanisms:
  Concepts:
  – Exceptions
- control operators such as callcc in Racket
- continuation passing style transformations

- Implementing a language.
  Concepts:
  - Define the abstract syntax using data type declarations.
  - Define the static semantics using the type system of the metalanguage.
  - Define the operational semantics using the operational semantics of the metalanguage.
  - Experiment with different scoping mechanisms and parameter passing techniques.

```c
void qsort(int a[], int lo, int hi) {
    int h, l, p, t;
    if (lo < hi) {
        l = lo;
        h = hi;
        p = a[hi];
        do {
            while ((l < h) && (a[l] <= p))
                l = l+1;
            while ((h > l) && (a[h] >= p))
                h = h-1;
            if (l < h) {
                t = a[l];
                a[l] = a[h];
                a[h] = t;
            }
        } while (l < h);
        a[hi] = a[l];
        a[l] = p;
        qsort(a, lo, l-1);
        qsort(a, l+1, hi);
    }
}
```

```
fun qsort [] = []
  | qsort (x::xs) =
    let
      (lt, ge) = List.partition (fn n => n < x) xs
      in
        (qsort lt) @ (x :: (qsort ge))
    end
```

Quicksort

Imperative Style
(C; Java would be similar)

Functional Style (SML)
Languages: You will learn the basic of these languages:

- ML (a statically typed, mostly functional language)
- Racket (a dynamically typed, mostly functional language)
- Haskell (a statically typed, pure functional language)
- Prolog (a logic programming language)

There are thousands of languages not on this list, many programming styles not represented, and many language constructs and concepts that it would be great to study.

Approximate Topic List:

1. Syntax vs. semantics vs. idioms vs. libraries vs. tools
2. ML basics (bindings, conditionals, records, functions)
3. Recursive functions and recursive types
4. Benefits of no mutation
5. Algebraic datatypes, pattern matching
6. Tail recursion
7. Higher-order functions; closures
8. Lexical scope
9. Currying
10. Syntactic sugar
11. Equivalence and effects
12. Parametric polymorphism
13. Type inference
14. Abstract types and modules
15. Racket basics
16. Dynamic vs. static typing
17. Laziness, streams, and memoization
18. Implementing languages, especially higher-order functions
19. Macros
20. Eval
21. Lazy evaluation
22. Object-oriented programming is dynamic dispatch
23. OOP vs. functional decomposition and extensibility
24. Programming with relations
25. Logic programming

Barriers and Accommodations: Our goal is a fully inclusive class, accessible to everyone. If you encounter or anticipate barriers to full participation and fair evaluation for any reason, please communicate your needs to the instructor so that we can find a suitable accommodation. You are also encouraged to contact the Accessible Education Center in 360 Oregon Hall at 541-346-1155 or uoaec@uoregon.edu. It is particularly important that you inform the instructor in the first week of the quarter if you require accommodation. Delayed notification of such a requirement may make it impossible to provide the accommodation.
Academic Honesty: The University Student Conduct Code (available at conduct.uoregon.edu) defines academic misconduct. Students are prohibited from committing or attempting to commit any act that constitutes academic misconduct. Students should properly acknowledge and document all sources of information (e.g., quotations, paraphrases, ideas) and use only the sources and resources authorised by the instructor. If there is any question about whether an act constitutes academic misconduct, it is the student's obligation to clarify the question with the instructor before committing to attempting to commit the act. Academic honesty is expected and cases of suspected dishonesty will be handled according to university policy. In particular, copying someone else's work (including material found on the web) will not be tolerated.

Academic Disruption due to Campus Emergency: In the event of a campus emergency that disrupts academic activities, course requirements, deadlines, and grading percentages are subject to change. Information about changes in this course will be communicated as soon as possible on Piazza. Students are encouraged to continue the readings and other assignments as outlined in this syllabus or subsequent versions of the syllabus.

Prohibited Discrimination and Harassment Reporting: I am a designated reporter. For information about my reporting obligations as an employee, please see Employee Reporting Obligations on the Office of Investigations and Civil Rights Compliance (OICRC) website. Students experiencing any form of prohibited discrimination or harassment, including sex or gender-based violence, may seek information and resources at safe.uoregon.edu, respect.uoregon.edu, or investigations.uoregon.edu. For contact the non-confidential Title IX office/Office of Civil Rights Compliance (541-346-3123), or Dean of Students offices (541-346-3216), or call the 24-7 hotline 541-346-SAFE for help. I am also a mandatory reporter of child abuse. Please find more information at Mandatory Reporting of Child Abuse and Neglect.