CS 3060 : PROGRAMMING LANGUAGES

Semester Hours: 3.0  
Contact Hours: 3

Coordinator: Sankardas Roy

Text: Seven Languages in Seven Weeks: A Pragmatic Guide to Learning Programming Languages

Author(s): BRUCE A. TATE

Year: 2010

SPECIFIC COURSE INFORMATION

Catalog Description:
Examination of a wide variety of programming languages, paradigms, features, and syntaxes through exposure to theory and hands on exercises. Topics covered include static, dynamic, strong and weakly typed, compiled and interpreted, object-oriented, functional and procedural programming, and decision constructs. Prerequisite: Grade of C or better in CS 2020.

Course type: REQUIRED

SPECIFIC COURSE GOALS

- I can identify differences and similarities across programming languages.
- I can describe the differences between different programming paradigms.
- I can implement basic algorithms using different programming paradigms.
- I can explain the strengths and weaknesses of different type systems.
- I can explain the differences between program compilation and interpretation.

COMPUTER SCIENCE STUDENT OUTCOMES ADDRESSED BY THIS COURSE

- CS 1 Analyze a complex computing problem and to apply principles of computing and other relevant disciplines to identify solutions
- CS 2 Design, implement, and evaluate a computing-based solution to meet a given set of computing requirements in the context of the program’s discipline
- CS 6 Apply computer science theory and software development fundamentals to produce computing-based solutions
LIST OF TOPICS COVERED

- Encapsulation, Polymorphism, Inheritance
- Higher Order Functions
- Purity (Side Effect Free)
- Immutability
- Lazy Evaluation
- Recursion
- Lambda Calculus
- Static & Dynamic Typing
- Garbage Collection
- Interpreted vs. Compiled Languages

COMPUTER SECURITY TOPICS

Faculty who recently offered CS 3060 have discussed and identified a list of topics related to computer security in this course. Below is a list for instructors to incorporate. (*) indicates topics are mandatory.

<table>
<thead>
<tr>
<th>Security Topic</th>
<th>Description</th>
<th>Textbook Reference</th>
<th>Estimated Class Hours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Security vulnerability due to side effects</td>
<td>Unexpected <em>side effects</em> can cause security vulnerability. We can construct an example showing <em>side effect</em> problem being present in an object-oriented programming language (e.g., Java) whereas a functional language like Haskell can help us control <em>side effect</em>.</td>
<td>No textbook. An example is available in Appendix</td>
<td>1</td>
</tr>
<tr>
<td>Type system as a security guard</td>
<td>Using a weakly typed language may lead to bugs or security issues. We can construct a few examples in C, C++, PHP, JavaScript, etc. On the other hand, Java is strongly typed. However, that does not mean it always leads to secure code.</td>
<td>Chapter 5. Figure 5-24 (available in Appendix)</td>
<td>2</td>
</tr>
</tbody>
</table>


A few concrete examples are available as Appendix which can be found in the folder hosting additional material related to Computer Security.