CS 301: Formal Languages and Functional Programming: Fall, 2016
Syllabus

1. **Logistics**.

   - Lectures: Monday, Tuesday, Wednesday, Friday; 8:00 in CF 226
   - Lab:
     - Section 42120: Tuesday, 12:00 – 1:50 in CF 416
     - Section 42121: Thursday, 2:00 – 3:50 in CF 416
   - Instructor: James Johnson
   - Office hours: CF 467, Monday, Wednesday, Friday, 9:00 – 10:00
   - Instructor contact.
     - e-mail: James.Johnson@wwu.edu
     - website: http://facultyweb.cs.wwu.edu/∼johnsoj5
   - Lab TA: Elizabeth Brooks, brookse5@wwu.edu
     - Office hours: MWF @ 10:00 and R @ 12:00 in CF 412.
   - Texts.
     - On-line books: (available for download from the course web page)
       * *Book of Proof*, by Richard Hammack, Virginia Commonwealth University, `http://cse.unl.edu/~choueiry/S11-235/files/BookOfProof.pdf`
       * *Introduction to the Theory of Computation* by Anil Maheshwari and Michiel Smid, School of Computer Science, Carleton University, `http://cg.scs.carleton.ca/~michiel/TheoryOfComputation`
     - Physical books:
       * *The Little Schemer*, 4$^{\text{th}}$ Edition, by Daniel Friedman and Matthias Felleison, MIT Press, 1996.
       * *Discrete Structures, Logic, and Computability*, 3$^{\text{rd}}$ edition, by James Hein, Jones and Bartlett Publishers, 2010. (**reference only, not required**)
   - Evaluation
     - There will be 25-minute quizzes, nominally six of them, throughout the term, comprising 30% of the final grade. There will be a final exam worth 40% of the final grade. The remaining 30% will be distributed over a number of laboratory assignments and Scheme programming assignments. Questions on the quizzes and on the final will deal only with concepts from the online texts. Programming assignments will deal primarily with material from The Little Schemer, although some may be based on concepts from the online texts
     - Tentative grade scale: 90+ = A, 85+ = A-, 80+ = B+, 75+ = B, 70+ = B-, 65+ = C+, 60+ = C, 55+ = C-, 50+ = D, 50- = F

2. **Course objectives**.

   - The course catalog description mentions several discrete mathematics concepts, such as sets, trees, functions, and relations. Our first course objective is to master these topics, together with various proof techniques. In this context, the first five weeks of the term will investigate the 13 chapters of *The Book of Proof*. As presented in the text, this material is too voluminous to consider in complete detail. Consequently, we will truncate most of the chapters, and you will be responsible only for specific sections of each chapter. These sections appear in the calendar on the course web page.
   - Also mentioned in the catalog description are formal language classes and their associated machine incarnations: regular languages and finite automata, context-free languages and pushdown automata, recursively enumerable languages and Turing machines. Our second objective is to acquire an introductory understanding of these topics, via Chapters 2–4 of the second on-line text, *Introduction to the Theory of Computation*. A more advanced treatment is available in the elective course, Automata and Formal Language Theory (CS 401).

- The final objective is to acquire introductory level competence in functional programming, specifically with the Scheme language. For this purpose, we will use the Racket programming environment, which is available on all the laboratory machines. A free download for your home machines is also available over the internet. All instruction in the Scheme language will occur in the weekly laboratory sessions.

3. How we plan to achieve these objectives.

   - *The Book of Proof* presents a collection of mathematical tools. This collection is cohesive, in the sense that all of the tools find use in reasoning about computation from a theoretical point of view. But the collection can appear rather disconnected because we learn to use the tools in isolated contexts, just rich enough to engage the usefulness of each tool. We never apply them in a coordinated fashion to solve a larger realistic problem. The tools provide various mechanisms for dealing with sets, relations, functions, logic, counting strategies, proof methods. Consequently, you are asked to solve a large number of small problems, rather than a few larger problems as is likely the case in your other courses.

     The web site contains a calendar that suggests a course through the material and enumerates a list of problems for homework. It is very important that you at least read through *all* of these homework problems. Many of them are very short, requiring only a few minutes of thought. Some serve only to provide practice with the commonly used notation associated with a particular tool. Some require more imagination and require that you extend the obvious applications of the concept under consideration. Most of you have full schedules, and it would be difficult for you to find the time to complete all of the problems. However, your understanding of the concepts will be proportional to the time spent solving problems.

     You are *not* required to submit these assignments. Consequently, you can use your own cryptic notation and handwriting that only you can read. Moreover, you are encouraged to form study groups to work on the problems. We will solve many of the homework problems in class, with priority given to problems suggested by class members. Some of the homework problems may appear in the quizzes, but only after there has been an opportunity to see a solution in class. Prior to any quiz, the relevant homework problem solutions will be available from the course web page.

   - *The Theory of Computation* narrows the focus to consider elementary computational models: finite automata, push-down automata, and Turing machines. Each of these models provides a context in which all of the tools find productive use. Consequently, there are fewer homeworks problems, although each problem typically involves the coordinated use of several of the tools learned in the first half of the term. We will continue to solve some of these problems in class, again with priority given to suggestions from class members.

   - In parallel with our progress through the texts, we will be studying the Scheme programming language as an example of how mathematical reasoning might be automated. The general idea in any programming language is to start with a few simple operations that can be combined in imaginative ways to develop more complicated tasks.

     In Scheme, the basic operation is recursion. Indeed, in the version that you will be learning in the lab, you will attempt to solve all of your assignments using primarily recursion, together with a very small collection of initial primitives. Some of the assignments will involve implementing operations on sets and functions as described in *The Book of Proof*. Others will involve problems that you would likely prefer to solve in a procedural language, such as Java. But, with a little thought, you will learn to use recursion in a more abstract fashion.

4. **Important dates**.

   - Monday, December 5, 2016, 1:00 – 3:00, CF 226: final exam.