

**Fayetteville State University**  
**College of Arts and Sciences**  
**Department of Mathematics and Computer Science**  
**CSC 207-01 – Symbolic Programming**  
**Fall 2010**

**I. Locator Information**

Instructor: Dr. Michael Almeida

Course # and Name: CSC 207-01 – Symbolic Programming

Day and Time Class Meets: TR 12:30 – 1:45 pm Room: SBE 231

Semester Credit Hours: 3

Office Hours: M 11:00 am – 4:00 pm;

TR 11:00 – 12:30 & by appointment

Total Contact Hours for Class: 45

Office Location: 340 SBE

Email address: malmeida@uncfsu.edu

Office Phone: 910-672-1161

**FSU Policy on Electronic Mail:** Fayetteville State University provides to each student, free of charge, an electronic mail account ([username@uncfsu.edu](mailto:username@uncfsu.edu)) that is easily accessible via the Internet. The university has established FSU email as the primary mode of correspondence between university officials and enrolled students. Inquiries and requests from students pertaining to academic records, grades, bills, financial aid, and other matters of a confidential nature must be submitted via FSU email. Inquiries or requests from personal email accounts are not assured a response. The university maintains open-use computer laboratories throughout the campus that can be used to access electronic mail.

Rules and regulations governing the use of FSU email may be found at

<http://www.uncfsu.edu/PDFs/EmailPolicyFinal.pdf>

Note: In case FSU must close for an emergency during the semester, instruction will continue using Blackboard.

**II. Course Description**

This course introduces the basic concepts and methods of symbolic programming. Symbolic programming involves the construction and analysis of complex symbolic expressions which can be used to represent many different types of information, including mathematical formulas, objects and their relations, and natural language sentences. This course also introduces functional programming and logic programming as two widely used paradigms for symbolic computation. Course topics include recursion, list processing, tree processing, backtracking, unification and resolution.

Prerequisites: Grade of “C” or better in CSC 130 and MATH 150

**III. Disabled Student Services**

In accordance with Section 504 of the 1973 Rehabilitation Act and the Americans with Disabilities Act (ACA) of 1990, if you have a disability or think you have a disability to please contact the Center for Personal Development in the Spaulding Building, Room 155 (1<sup>st</sup> Floor); 910-672-1203.

**IV. Textbooks**

In this course the Blackboard notes are primary while the textbooks are optional.

Dybvig, R. Kent (2003) *The Scheme Programming Language, 3<sup>rd</sup> Edition*. MIT Press.

A free electronic version of this textbook is available online at: [www.scheme.com/tspl3/](http://www.scheme.com/tspl3/)

Bratko, Ivan (2001) *Prolog Programming for Artificial Intelligence, 3<sup>rd</sup> Edition*. Addison-Wesley.  
(recommended, but any other Prolog book will do)

Scheme, a dialect of Lisp, will be used as the functional programming language, and Prolog will be used as the logic programming language. Implementations of these programming languages are available in the CS labs in SBE and online.

*PLT Scheme* (now called *Racket*) can be downloaded free at: <http://racket-lang.org/download/>

*MIT/GNU Scheme* can be downloaded free at: [www.gnu.org/software/mit-scheme/](http://www.gnu.org/software/mit-scheme/) (the IDE isn't as good as that of PLT Scheme but it's ok)

*SWI-Prolog* can be downloaded free at: <http://www.swi-prolog.org/download/stable> (be sure to select **.pl** as the extension when installing)

## V. Student Learning Outcomes

Upon completion of this course, students will be able to:

1. write procedures that construct and manipulate symbolic expressions;
2. write recursive procedures that perform numerical and list operations;
3. write tail recursive procedures;
4. write programs in the functional programming paradigm;
5. implement recursive rules in a logic programming language;
6. write programs in the logic programming paradigm;
7. use negation-as-failure in programs;
8. describe the matching algorithm.

## VI. Course Requirements and Evaluation Criteria

a. Final grades are assigned as follows:

A: 90-100

B: 80-89

C: 70-79

D: 60-69

F: < 60

- b. Attendance Requirements – Students are expected to attend class regularly and to complete the in-class exercises. The in-class exercises prepare you to do the programming projects.
- c. The course grade is based on six programming projects (10% each for 60%), a midterm exam (15%), a final exam (15%) and in-class exercises (10%).
- d. Policy on Late Assignments - Each assignment must be submitted on time. Five points will be deducted from your assignment grade for each school day the assignment is overdue. No assignment will be accepted more than one week after the due date without an excellent excuse. Please contact me before the deadline if you won't be able to submit something on time.
- e. Dishonesty on graded assignments will not be tolerated. Although students may discuss assignments with one another, they must neither give nor receive excessive help. Students learn by doing things themselves. Having access to another student's work on the system is definitely not allowed. Duplicate answers are not acceptable. Each student is responsible for disposing of printouts safely (Do NOT simply throw away printouts in a trash can where they can easily be retrieved by another person.) and for protecting their home directory. All students involved in dishonesty (those giving as well as those receiving unallowable help) will be penalized.

Please note: If these evaluation criteria must be revised because of extraordinary circumstances, the instructor will distribute a written amendment to the syllabus.

## VII. Academic Support Resources – none for this course.

## VIII. Course Outline and Assignment Schedule\*

<b>week #</b>	<b>start date</b>	<b>readings</b>	<b>assignments</b>	<b>events</b>
1	19-Aug	Course overview; Intro to Scheme; Running Scheme	download & install Scheme	Classes start 8/19
2	23-Aug	Intro to Scheme; Scheme lists	Project #1 assigned	
3	30-Aug	Scheme procedures		
4	6-Sep	Conditionals; Functional programming 1	Project #2 assigned	Monday holiday 9/6
5	13-Sep	Intro to recursion; Recursion on numerical functions		
6	20-Sep	Symbolic programming; Tracing in Scheme; Recursion on simple list functions	Project #3 assigned	Fall Convocation 9/21
7	27-Sep	Recursion on list constructing functions; Functional programming 2		
8	4-Oct		Midterm Exam 10/7	Midterms start 10/7
9	11-Oct	Double recursion; Trees in Scheme; Tail recursion	Project #4 assigned	Fall Break 10/15-18
10	18-Oct	Intro to Prolog; Running Prolog	Download & install prolog;	
11	25-Oct	Prolog syntax; Tracing in Prolog		
12	1-Nov	Prolog arithmetic; Prolog lists	Project #5 assigned	
13	8-Nov	Backtracking & Negation		Holiday 11/11
14	15-Nov	Binary Trees	Project #6 assigned	
15	22-Nov	Resolution		Holiday 11/25-26
16	29-Nov	Review		Classes end 12/3
17	6-Dec		Final Exam 12/7	

\* This schedule is subject to change for the optimum benefit of the class as a whole. Therefore it is important to stay alert and attend class regularly.

## **IX. Teaching Strategies**

The primary teaching strategies for this course will be lectures, supplementary textbook readings, in-class exercises, and programming projects.

## **X. Bibliography**

Bramer, Max. *Logic Programming with Prolog*. Springer, 2005.

Clocksin, William F. & Christopher S. Mellish. *Programming in Prolog, 5<sup>th</sup> Edition*. Springer, 2003.

Friedman, Daniel P. & Felleisen, Matthias. *The Little Schemer, 4<sup>th</sup> Edition*. MIT Press, 1996.

Harvey, Brian & Wright, Matthew. *Simply Scheme*. MIT Press, 1994.