UNIVERSITY OF NEBRASKA AT OMAHA
COURSE SYLLABUS/DESCRIPTION

Department and Course Number	CSCI 4220
Course Title	Principles of Programming Languages
Course Coordinator	Victor Winter
Total Credits	3
Date of Last Revision	February, 2014

1.0 Course Description Information:

1.1 Catalog description:

2 This course covers the foundational concepts and principles underlying the design and implementation of programming languages. Language constructs including assignment, equality, references, aggregations, scope, encapsulation, and parameter passing are discussed. A central theme is how a particular language construct relates to the concept of equational reasoning (referential transparency). Formal notations for describing syntax and semantics are presented.

2.1 Prerequisites of the course:

2.1.1 Data structures and algorithms (CSCI 3320)
2.1.2 Introduction to the Theory of Computation (CSCI 3660)

2.2 Overview of content and purpose of the course: *This section provides more detail than the catalog description.*

Programming languages are the mechanism though which computer technology is leveraged. In order to more effectively manage the scale and complexity of modern applications, programming languages have become quite sophisticated along the axes of: syntax, type systems, and semantics. In practice, the features of modern languages can be more effectively integrated into the developers skill set if the developer has an understanding of the foundational concepts from which those features evolved. Modern software development also demands that the automated capabilities of tools be fully leveraged. In particular, the developer needs to have a clear understanding of how tools contribute to the development of a software system. The basis of this understanding comes from the theoretical limits of what can and can not be computed. For example, the type system of a programming language can (automatically) catch only certain kinds of errors.

The purpose of this course is to lay a foundation for the knowledge of how to effectively use programming languages and tools to meet the needs of modern software development.

2.3 Unusual circumstances of the course.

There are no unusual circumstances for this course.

2.0 Course Justification Information

2.1 Anticipated audience / demand:

This course is intended for junior/senior students in computer science and related disciplines interested in studying the foundational concepts underlying programming languages.

2.2 Indicate how often this course will be offered and the anticipated enrollment:

This course will be offered every semester. The anticipated enrollment is 25 students per semester.

2.3 If it is a significant change to an existing course, please explain why it is needed:

NA

3.0 List of performance objectives stated in learning outcomes in a student's perspective:

By completing this course students will be expected to understand:

- **Language Design Principles**. To appreciate the intimate relationship between language and thought and why therefore a well-designed language must conform to certain principles
- **Syntax**. To be able to read and write formal notations describing the syntax of a programming language
- **Semantics.** To develop a more precise understanding of language semantics as well as the issues and trade-offs underlying various language design decisions (e.g., mutable versus immutable values).
- **Types.** To develop a rudimentary appreciation of the contribution of type systems to language design.
- **Language Constructs.** To understand how various language constructs relate and differ from one another both within a single language as well as across languages (e.g., parameter passing and assignment).
- **Language Paradigms.** To develop a basic understanding of the similarities and differences among a variety of common language paradigms (e.g., imperative, object-oriented, functional, declarative). Students will be expected to write programs in non-imperative language paradigms.
- **Experience and Ethics.** To become better prepared culturally to succeed in the workplace.

4.0 Content and Organization Information

4.1 List the major topics central to this course:

- Background and Motivation (3 hours)
  - Language and thought
  - The limits of computation
  - Language Design Principles
  - Brief history of programming languages
- Syntax (9 hours)
  - Context-free grammars
  - Regular expressions

    - Analysis
- Semantics (16 hours)
    - Equational Reasoning and/or Logic
    - Denotational semantics and/or other frameworks
- Language Paradigms (15 hours)
    - A subset of the following paradigms will be discussed:
    - Functional
    - Imperative
    - Object-oriented
    - Declarative
    - Logic programming
    - Concurrent
- Experience and Ethics (2 hours)
    - Material from this category will be used to motivate and round out the technical material covered in the course.

5.0 Teaching Methodology Information

5.1 Methods:

The course format is predominantly based on lectures. However, students will be expected to participate in discussions of the various topics as they are studied.

5.2 Student role in the course.

Students are expected to attend lectures, participate in class discussions, and complete assignments in a timely fashion.

5.3 Contact hours.
3 hours per week.

6.0 Evaluation Information

6.1 Describe the typical types of student projects that will be the basis for evaluating student performance:

This course is exclusively an undergraduate course. The following artifacts are produced by students:

1. Homework and programming assignments
2. Blackboard quizzes
3. In-class exams
4. A group project
5. Extra credit problems and projects

6.2 Describe the typical basis for determining the final grade (e.g. weighting of various student projects):

The weighting between in-class exams and homework may vary between sections of the course. Shown below is an example of the general weighting that may be used.

40% In-class exams
40% Homework and programming assignments
20% Project

Tentatively, exams are scheduled every seven weeks.

6.3 Grading type:

Letter grades

The grading scale will be determined by the instructor and communicated to students at the start of the term. What follows is one possible example.

$90 \leq A < 97 \leq A+ \leq 100$
$80 \leq B < 87 \leq B+ < 90$
$70 \leq C < 77 \leq C+ < 80$
$60 \leq D < 67 \leq D+ < 70$
$0 \leq F < 60$

7.0 Resource Material Information

Textbooks and/or other required readings used in course:.

[1] A. Tucker and R. Noonan. *Programming Languages: Principles and Paradigms (second edition)*. McGraw Hill.

[2] J. D. Ullman. *Elements of ML Programming*. Prentice Hall.

7.1 Other suggested reading materials, if any.
Individual instructors will assign other materials as relevant to each course.

URL's of various documents and tools will be given to students on an as needed basis.

7.2 Current bibliography of resource for student's information (At least 10)

[1] T. W. Pratt and M. V. Zelkowitz. *Programming Lanugages: Design and Implementation.* Prentice Hall.

[2] R. W. Sebesta. *Concepts of Programming Languages*. Addison Wesley.

[3] K. C. Louden. *Programming Languages: Principles and Practice*. Thomson Brooks.

[4] R. Sethi. *Programming Languages:Concept and Construct*. Addison Wesley.

[5] C. Ghezzi and M. Jazayeri. *Programming Language Concepts.* John Wiley & Sons.

[6] M. R. Scott. *Programming Language Pragmatics.* Morgan Kaufmann Publishers.

[7] A. Fischer F. and Grodzinsky. *The Anatomy of Programming Languages*. Prentice Hall.

[8] M. Sipser. *Introduction to the Theory of Computation.* Course Technology.

[9] C. Meyers, C. Clack, and E. Poon. *Programming with Standard ML*. Prentice-Hall.

[10] L. C. Paulson. *ML for the Working Programmer*. Cambridge University Press.

[11] R. J. Schalkoff. *Programming Languages and Methodologies*. Jones and Bartlett Publishers International.

[12] P. Linz. *An Introduction to Formal Languages and Automata*. 5th ed. Jones and Bartlett Publishers International.

8.0 Estimate Computer Science Accreditation Board (CSAB) Category Content (class time in hours):

CSAB Category	*Core*	*Advanced*
Data structures	0	0
Computer organization and architecture	0	0
Algorithms and software design	5	0
Concepts of programming languages	40	0

9.0 Oral and Written Communications:

Every student is required to submit at least __0__ written reports (not including exams, tests, quizzes, or commented programs) to typically ____ pages and to make ____ oral presentations of typically __ minutes duration. Include only material that is graded for grammar, spelling, style, and so forth, as well as for technical content, completeness, and accuracy.

10.0 Social and Ethical Issues:

Topics relating to social and ethical issues are covered in an ad hoc manner. The purpose of discussing such issues is to help students succeed in the workplace.

11.0 Theoretical content

	Contact hours
Computability and notation	1.0

Context-free grammars	7.0
Regular expressions	5.0
Unification	1.5
Equational reasoning	1.0
Models of computation	2.0
Semantics	6.0
Logic	4.5

12.0 Problem analysis

Students learn the syntactic and semantic principles behind modern programming languages. They learn to analyze program structure and behavior in a formal setting and acquire sufficient depth to reason about program fragments.

13.0 Solution design

Students acquire an understanding of the rationale behind the design of a given model of computation (e.g., an environment function and a store function). Several models (e.g., symbol table vs environment/ store function) are contrasted and strengths and weaknesses are discussed.

CHANGE HISTORY

Date	***Change***	***By whom***	***Comments***
08/27/2002	Initial ABET version	Winter	
06/13/2003	Cleanup	Wileman	
12/11/2008	Update	Winter	
12/11/2008	Mapping table added	Winter	
05/24/2011	Revise the format of the syllabus for CCMS	Winter	Added entries to the bibliography. Revised content of several sections.
02/17/2014	Reviewed document	Winter	No changes recommended

UNIVERSITY OF NEBRASKA AT OMAHA
Mapping of CS Program Outcomes vs. course objectives

Department and Course Number	CSCI 42200
Course Title	Principles of Programming Languages
Course Coordinator	Victor Winter
Total Credits	3
Date of Last Revision	May 24, 2011

Instructions: Paste or type the course objectives in the left-hand column. Indicate the relationship between course objective and program outcome by placing one of the two following marks in the appropriate cell:

S – Strong relationship
X – Contributing relationship

Course objective	**CS Program Outcomes**										
	(a) knowledge of discipline	**(b) analyze problem, define**	**(c) design and implement solution**	**(d) function on a team**	**(e) ethical issues**	**(f) communicate effectively**	**(g) analyze impact of computing**	**(h) continued professional development**	**(i) Current techniques and tools**	**(j) apply foundations**	**(k) apply design and development**
1. Language Design Principles.	**X**									**X**	
2. Syntax	**S**	**X**	**X**	**X**					**X**	**X**	**X**
3. Semantics	**S**	**X**	**X**	**X**						**S**	**S**
4. Types	**X**	**X**	**X**	**X**						**X**	**X**
5. Language Constructs	**X**										
6. Language Paradigms	**X**										
7. Experience and Ethics					**X**						

Student Work	CS Program Outcomes										
	(a) knowledge of discipline	(b) analyze problem, define requirements	(c) design and implement solution	(d) function on a team	(e) ethical issues	(f) communicate effectively	(g) analyze impact of computing	(h) continued professional	(i) Current techniques and tools	(j) apply foundations	(k) apply design and development principles
Programming Assignments	**S**		**S**							**S**	**X**
Quizzes	**X**									**X**	
Exams	**X**		**X**							**X**	**X**
Project	**S**		**S**	**X**						**S**	**X**

CS Program Outcomes (2008)

(a) An ability to apply knowledge of computing and mathematics appropriate to the discipline;
(b) An ability to analyze a problem, and identify and define the computing requirements appropriate to its solution;
(c) An ability to design, implement and evaluate a computer-based system, process, component, or program to meet desired needs;
(d) An ability to function effectively on teams to accomplish a common goal;
(e) An understanding of professional, ethical, legal, security, and social issues and responsibilities
(f) An ability to communicate effectively with a range of audiences
(g) An ability to analyze the local and global impact of computing on individuals, organizations and society
(h) Recognition of the need for, and an ability to engage in, continuing professional development
(i) An ability to use current techniques, skills, and tools necessary for computing practices
(j) An ability to apply mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems in a way that demonstrates comprehension of the tradeoffs involved in design choices
(k) An ability to apply design and development principles in the construction of software systems of varying complexity.