1.0 Course Description:

1.1 Overview of content and purpose of the course (Catalog description).

This course covers the foundational concepts and principles underlying the design 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.

1.2 For whom course is intended.

This course is primarily for senior-level undergraduate students.

1.3 Prerequisites of the course (Courses).

CSCI 3320, CSCI 3660

1.4 Prerequisites of the course (Topics).

1.4.1 Data structures and algorithms.
1.4.2 Two semesters of programming in a high-level language such as C++ or Java.
1.4.3 Regular expressions and context-free grammars.
1.4.4 Basic concepts from discrete mathematics including set theory and first-order logic.

1.5 Unusual circumstances of the course.

None

2.0 Objectives:

List of performance objectives stated in terms of the student educational outcomes.

2.1 Language Design Principles. To appreciate the intimate relationship between language and thought and why therefore a well-designed language must conform to certain principles.

2.2 Syntax. To be able to read and write formal notations describing the syntax of a programming language, and to be able to understand why certain classes of language grammars work with certain types of processing techniques.
2.3 **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).

2.4 **Types.** To develop a rudimentary appreciation of the contribution of type systems to language design.

2.5 **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).

2.6 **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.

2.7 **Experience and Ethics.** To become better prepared culturally to succeed in the workplace.

### 3.0 Content and Organization:

List of major topics to be covered in chronological sequence.

3.1: Background and Motivation (3 hours)
   3.1.1: Language and thought
   3.1.2: The limits of computation
   3.1.3: Language Design Principles

3.2: Syntax (9 hours)
   3.2.1: Context-free grammars
   3.2.1: Regular expressions
   3.2.1: Analysis

3.3: Semantics (16 hours)
   3.3.1: Equational Reasoning and/or Logic
   3.3.1: Denotational semantics and/or other frameworks

3.4: Language Paradigms (15 hours)
   3.4.1: A subset of the following paradigms will be discussed:
   - Functional
   - Imperative
   - Object-oriented
   - Declarative
   - Logic programming
   - Concurrent

3.5: Experience and Ethics (2 hours)
   3.5.1: Material from this category will be used to motivate and round out the technical material covered in the course.
Please reference the link below to review lecture notes and quizzes to use in studying for the upcoming CSCI 4220 waiver exam:

http://faculty.ist.unomaha.edu/winter/4220-LectureNotes.zip