Course Outline

1. Document Information

Degree Program	Computer Science
Course Number	CS 311
Course Title	Theory and Implementation of Programming Languages
Semester Hours	3
Course Coordinator	Norman Carver
Revision Term	Spring 2020
Latest Revision	Spring 2020

2. Catalog Description

Introduction to the theory and implementation of programming languages including finite automata, regular grammars, lexical analysis, parsing, syntax-directed translation, semantic analysis, binding variables, data types, static and dynamic scope, subprograms, abstraction, and concurrency. Study of object-oriented, functional, and logic programming languages. Lab work is essential.

3. Textbooks

- Sebesta, R.W. (2019). Concepts of Programming Languages, Pearson, 12th Edition. ISBN: 9780134997186.

4. References

- Kirk, D.B. & Hwu, W-M. (2017). Programming Massively Parallel Processors: A Hands-on Approach, Elsevier, 3rd Edition. ISBN: 978-0128119860.
- Gropp, W., Lusk, E. & Skjellum, A. (2014). Using MPI: Portable Parallel Programming with the Message-Passing Interface, MIT, 3rd Edition. ISBN: 978-0262527392.

5. Course Learning Outcomes

- To obtain background on compilers and language compilation
- To understand the basics of the theory of computing applied to develop programming languages

- To learn the features and capabilities those are available in programming languages
- To understand the issues in implementing various programming language features
- To learn the effect of languages on problem solving and programming process.

6. Assessment of the Contribution to Student Outcomes

Outcome	1	2	3	4	5	6
Assessed	X	X				X

7. Prerequisites by Topic

CS 220 with a grade of C or better.

8. Major Topics Covered in the Course

1. Introduction: domains, language evaluation criteria, language categories, implementation methods {3 classes}
2. Syntax and semantics: formal methods of describing syntax, attribute grammars, dynamic semantics {6 classes}
3. Finite automata: deterministic and nondeterministic finite automata, regular grammars {5 classes}
4. Lexical and syntax analysis: recursive-descent parsing, bottom-up parsing {5 classes}
5. Variables: names, binding, types, scope, lifetime {2 classes}
6. Basic data types: implementations of integers, strings, etc. {2 classes}
7. Expressions: operators, assignment, precedence, associatively, side effects, overloading, coercion {2 classes}
8. Subprograms: procedural abstraction, generic functions, parameter passing, recursion {2 classes}
9. Abstract data types: data abstraction, user-defined data types, encapsulation, information hiding {2 classes}
10. Concurrency: monitors, threads {2 classes}
11. Exception and event handling {2 classes}
12. Object-oriented programming: basic features, alternative models, implementation requirements {3 classes}
13. Functional and logic programming: clips, lisp, scheme {4 classes}