|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| AI3XX – Structure and Interpretation of Computer Programs **(3 CH) – Tentative Syllabus** | | | | | | | |
| Course Code: **AI3XX** Title: **SICP**  Instructor: **Taj Muhammad Khan** Office hours: ------- **TueWedThu**  Credit Hours: **3** Lecture Hours: **3** Lab Hours: **0** Pre-Req: **CS221** Co-Req: **None** Profile: **WK1** | | | | | | | |
| **Course Introduction** | | | | | | | |
| Computer programs underpinnings can be said to have two founts: the Turing Machine and the Lambda Calculus. While the former is covered in most undergraduate courses dealing with programming, the coverage for the latter, Lambda Calculus, is much rare even though it is at the foundations of Functional Programming. Based on one of the most famous books in CS, this course will look at computer programming from the Lambda Calculus side of things including algorithmic thinking, computational complexity, and programming models. It goes through the foundation of functional programming, including higher order functions, then moves to data abstraction concepts, including type systems, and mutability. | | | | | | | |
| **Course Contents** | | | | | | | |
| * Intro to Functional Programming. * Evaluation models, Computational Processes. * Higher Order Functions * Encapsulation * Generic Functions | | | | * Types and Type Systems * State * Modularity * Objects and Mutable State * Working of Scheme | | | |
| **Mapping of CLOs and PLOs** | | | | | | | |
| **Sr. No** | | **Course Learning Outcomes (CLOs) †** | | | **PLOs \*** | | **Blooms Taxonomy** |
| CLO-1 | | Demonstrate the ability to write programs in functional programming languages. | | | PLO1 | | C3 (Apply) |
| CLO-2 | | Apply functional programming concepts in analyzing a problem and formulating its solution. | | | PLO2 | | C3 (Apply) |
|  | | **†**  Please add the prefix “Upon successful completion of this course, the student will be able to”  **\***PLOs are for Engineering programs only | | | | | |
| **CLO Assessment Mechanism** | | | | | | | |
| **Assessment Tools** | | | **CLO-1** | **CLO-2** | |  | |
| **Assignments** | | | 100% | - | | - | |
| **Projects** | | | - | 100% | | - | |
|  | | | - | - | | - | |
|  | | | - | - | | - | |
| **Overall Grading Policy** | | | | | | | |
| **Assessment Items** | | | | **Percentage** | | | |
| Assignments | | | | 10% | | | |
| Quizzes | | | | 10% | | | |
| Project | | | | 15% | | | |
| Mid-term Exam | | | | 25% | | | |
| Final Exam | | | | 40% | | | |
|  | | | |  | | | |
| **Books and Materials** | | | | | | | |
| **Textbooks:**   * Structure and Interpretation of Computer Programs - 2nd Edition by Harold Abelson, Gerald Jay Sussman.   **Resources**:   * [Book is freely available] (<http://sarabander.github.io/sicp/>) * [Abelson and Sussman lectures] (<https://www.youtube.com/playlist?list=PLE18841CABEA24090>) * [Brian Harvey cs61a lectures] (<https://www.youtube.com/playlist?list=PLhMnuBfGeCDNgVzLPxF9o5UNKG1b-LFY9>) | | | | | | | |
| **Administrative Instructions** | | | | | | | |
| * Course contents and their order may be slightly modified during the course execution. * According to institute policy, 85% attendance is *mandatory* to appear in the final examination. * Assignments must be submitted as per instructions and by the deadline. Unannounced quizzes in case of low attendance. * In any case, there will be no retake of (scheduled/surprise) quizzes or exams. * Disciplinary action may be taken for any kind of malpractice, e.g., plagiarism, cheating, misbehavior in class, etc. * For queries, kindly follow the office hours to avoid inconvenience. Take appointment otherwise. Email is the official channel of communication. Will not respond to other channels. * Read the book. | | | | | | | |
| **Lecture Breakdown** | | | | | | | |
| **Week #** | Topics | | | | | | |
| **Week 01** | Introduction to Course. | | | | | | |
| **Week 02** | Procedural Abstraction: Foundations of functional programming. | | | | | | |
| **Week 03** | Procedural Abstraction: Introduction to Scheme and evaluation models. | | | | | | |
| **Week 04** | Procedural Abstraction: Computational Processes. | | | | | | |
| **Week 05** | Procedural Abstraction: Higher order functions. | | | | | | |
| **Week 06** | Data Abstractions: Encapsulation. | | | | | | |
| **Week 07** | Data Abstractions: Generic Functions. | | | | | | |
| **Week 08** | Data Abstractions: Types. | | | | | | |
| **Week 09** | Data Abstractions: Type Systems. | | | | | | |
| **Week 10** | Modularity, Objects, and State: Decomposition. | | | | | | |
| **Week 11** | Modularity, Objects, and State: Objects and Mutable State. | | | | | | |
| **Week 12** | Modularity, Objects, and State: Delayed Evaluation. | | | | | | |
| **Week 13** | Modularity, Objects, and State: Streams. | | | | | | |
| **Week 14** | (Time permitting) Build an interpreter for scheme. | | | | | | |
| **Week 15** | (Time permitting) Build an interpreter for scheme. | | | | | | |