**COURSE INFORMATION SHEET**

**(For Theory + Lab Based Course)**

|  |  |
| --- | --- |
| **Session:** | Spring-2021 |
| **Course Title:** | Compiler Construction |
| **Course Code:** | CS-310 |
| **Credit Hours:** | 3+1 |
| **Semester:** | 5th |
| **Pre-Requisites:** | Finite Automata Theory |
| **Instructor Name:** | Ayesha Urooj , RajKumar ,Raheel Usman |
| **Email and Contact Information:** | [aurooj@ssuet.edu.pk](mailto:aurooj@ssuet.edu.pk), [raj.chawla@ssuet.edu.pk](mailto:raj.chawla@ssuet.edu.pk), [rausman@ssuet.edu.pk](mailto:rausman@ssuet.edu.pk) |
| **WhatsApp Group** | Compiler Const\_2019 |
| **Office Hours:** | *09:00am – 05:00pm* |
| **Mode of Teaching:** | Synchronous/Asynchronous/ Hybrid/Blended |

**COURSE OBJECTIVE:**

* Provide an understanding of the fundamental principles in Compiler Construction.
* Learn the process of mapping and design the self-developed language to implement different parsing techniques inorder to make efficient parsers.
* Provide the skills needed for building compilers for various situations that one may encounter in a career in Computer Science.

**COURSE OUTLINE:**

Introduction to Interpreter and compilers, Compiler Techniques and Methodology. Organization of Compliers. Lexical and Syntax Analysis .Parsing techniques, Types of parser, Type checking, Semantic analyzer, Object code generation and optimization, detection and recovery from errors . Contrast between compilers and interpreter.

**COURSE LEARNING OUTCOMES (CLOs) and its mapping with Program Learning Outcomes (PLOs):**

|  |  |  |  |
| --- | --- | --- | --- |
| **CLO No.** | **Course Learning Outcomes (CLOs)** | **PLOs** | **Bloom’s Taxonomy** |
| CLO 1 | **Recall** the Concepts of Languages ,R.Es and F.As. |  | C1(Remember) |
| CLO 2 | **Discuss** the fundamental concepts of the basic structure of compiler and recall R.E and FA. |  | C2(Understand) |
| CLO 3 | **Apply** different methodsand techniques through which one can design its own Language. |  | C3(Applying) |
| CLO 4  (Lab+Theory) | **Analyze** the Different parsers and draw connections between them |  | C4(Analyze) |
| CLO 5  (Lab+Theory) | **Design** and **Map** the parser of their own generated Grammar in any host Language. |  | C6(Creating) |

**COMPLEX ENGINEERING PROBLEM/ACTIVITY:**

|  |  |
| --- | --- |
| **Complex Engineering Problem Details** | **Included: NO** |
| **Complex Engineering Activity Details** | **Included: NO** |

**RELATIONSHIP BETWEEN ASSESSMENT TOOLS AND CLOs:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Assessment Tools** | **CLO-1**  **(6 )** | **CLO-2**  **(6)** | **CLO-3**  **(21)** | **CLO-4**  **(32)** | **CLO-5**  **(35)** |
| **Quizzes** |  | 16.66% (1) |  | 6.25% (2) | 5.71% (2) |
| **Assignments** | 16.66% (1) |  | 4.76%(1) |  | 8.57% (3) |
| **Midterm Exam** | 83.33% (5) | 83.33% (5) | 47.61% (10) |  |  |
| **Final Exam** |  |  | 47.61% (10) | 62.5% (20) | 57.14% (20) |
| **Lab Assessment** |  |  |  | 31.25% (10) | 28.57% (10) |

**GRADING POLICY:**

|  |  |
| --- | --- |
| **Assessment Tools** | **Percentage** |
| Quizzes | 5% |
| Assignments | 5% |
| Midterm Exam | 20% |
| Final Exam | 50% |
| Lab Assessment | 20% |
| **TOTAL** | **100%** |

**Recommended Book:**

* Alfred V. Aho, Ravi Sethi, Jeffrey D. Ullman Compilers: Principles, Techniques, and Tools, 2nd Edition, Boston Pearson Education, Inc.

**Reference Books:**

* Allen Holub ,Compiler Design in C, Latest edition,Prentice-Hall, Inc. New Jersey

**COURSE BREAKDOWN WITH LAB SYNCHRONIZATION:**

**- Both sides same Colours:** Lab is synchronized with the topic

**- Red Color:** Lab is not synchronized (*conducted before theory*)

- **No Color:** Lab is to introduce new hardware or software skill **/**

Open Ended Lab / Lab is relevant to a topic taught inpre-

requisite and required for upcoming labs

|  |  |  |
| --- | --- | --- |
| **Week No.** | **Topics** | **Laboratory Synchronization** |
| 1 | Translators,  History of Compilers,  Language processing system,  Types of Languages and compilers,  Compiler Phases with Block diagrams | Overview of Compilers with e.g and explore the IDE by Making simple program which recognizes the key strokes as you press different letters. |
| 2 | Accessories of compiler ,  Symbol table management ,  cousin of compiler Pre Processor , Assembler etc | This Lab gives the flavor that how we can manage the lexemes in a symbol table by the concept of filing. |
| 3 | Introduction to context free grammar terminals ,  non-terminals , production rules ,  Grammar for Identifiers , Variable | Write a lexical analyzer in any language for the given tokens |
| 4 | Grammar for Multiple data type declaration, basic arithmetic operations,  Loops,  If else Statements,input, Buffering Technique | This lab covers some efficiency issues concerned with the buffering of input. First we’ll look at the two-buffer input scheme that is useful when look-ahead on the input is necessary to identify the tokens |
| 5 | Role, Features, Lexeme, Pattern, Token, Considerations for a simpler design of LA, ways of implementation | This lab, we’ll explore a useful technique for speeding up the lexical analyzer, i.e. the use of ***sentinels*** called **Buffering Technique II** |
| 6 | Specification and recognition of Tokens and implementation through systems programming languages | This lab is designed in order that how the tokens and Identifiers get recognition in a given input string. |
| 7 | Natural Language Grammar, formal grammar, context-free grammar, processing of grammars | Designing of context-free grammar for your own language. It must cover Multiple declaration, Loops, conditional statements User define functions. |
| 8 | **Mid-Term Week** | |
| 9 | Ambiguous and unambiguous grammar of expressions and if statement | Implementation of Lexical Analyzer in any programming language for **your context-free grammar.**  The steps are:   1. Identification of Tokens 2. Specification of Tokens in terms of Regular Expressions 3. Recognition of Tokens in terms of Transition diagrams 4. Coding of Lexical analyzer |
| 10 | Left recursion , left factoring, simple and fast implementation of predictive parsers, with examples | Transform your Grammar in the unambiguous form and perform the left recursion and left factoring from the grammar. |
| 11 | First and Follow sets, parsing table and moves of predictive parser with examples | Find out the First () and Follow () of the grammar which you have completed in the previous lab. |
| 12 | implementation of Non Recursive Predictive Parser with its stack implementation | Implement the Non recursive predictive parser by mapping the functions of First () & Follow () and making the Parsing table, perform the stack implementation by taking any program of your own language and shows its output. |
| 13 | Shift-reduce parsers , handles and stack implementation with examples ,Canonical collection, parsing table and moves made by the LR parser with examples | Transform Your Grammar in Augmented form in order to Implement SLR parser and find the closure and Goto operation. |
| 14 | Shift Left Right Parser, Stack implementation with table and examples. | Design an algorithm for SLR parser and constructing the Transition Diagram of the complete set of all the closures. |
| 15 | Syntax Directed Translation, Inheritance, synthesized attributes, dependency graph, | Write the complete code of SLR parser with the help of making Transition diagram, also show its ouput. Take any program of your own language as an input and display the output |
| 16 | Translation scheme, type checking of statements and expressions. |  |

**LECTURE PLAN**

**Course Title:**

**Course Code:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Week No.** | **Week Dates** | **Topics** | **Required Reading** | **Key Date** |
| **1** | 15-02-2021 to  19-02-2021 | Introduction to compilers  Translators, features of compiler, History of Compilers, Analysis of source program, Language processing system, Types of Languages and compilers | AA -Chap 1:  pg. 01 – 10  AH-Chap 1:  pg. 1-6 |  |
| **2** | 22-02-2021 to  26-02-2021 | Compiler Phases with Diagrams  Accessories of compiler  Symbol table management  cousin of compiler Pre Processor , Assembler | AA -Chap 1:  pg. 10 – 16 |  |
| **3** | 01-03-2021 to  05-03-2021 | Introduction to context free grammar terminals, Non-Terminals  Production Rules  Grammar for Identifiers , Variable Declaration | AH-Chap 1:  pg 7,8  AH-Chap 2:  pg 52 & 53 | Assignment # 1 |
| **4** | 08-03-2021 to  12-03-2021 | Grammar for Multiple data type declaration  basic arithmetic operations, Loops  If else Statements | AH-Chap 2:  pg 54  AA-Chap 2:  Pg. 26 , 27,32 | **Quiz#1** |
| **5** | 15-03-2021 to  19-03-2021 | Role, Features, Lexeme, Pattern, Token,  Considerations for a simpler design of LA, Input Buffering Technique  ways of implementation of LA | AA-Chap 3:  Pg. 84-88 |  |
| **6** | 22-03-2021 to  26-03-2021 | Specification and recognition of Tokens  Implementation of Tokens  Implementation of Tokens Examples | AA-Chap 3:  Pg 88-94 | Assignment # 2 |
| **7** | 29-03-2021 to  02-04-2021 | Natural Language Grammar  formal grammar, context-free grammar  Processing of grammars | AA-Chap 3:  Pg 105,168-170  AH-Chap 2  Pg 52-54  AH-Chap 3:  Pg 166 |  |
| **8** | **Midterm Examination**  **(05-04-2021 to 10-04-2021)** | | | |
| **9** | 12-04-2021 to  16-04-2021 | Ambiguous in Grammar  unambiguous grammar  Example of Ambiguous and unambiguous | AA-Chap 4:  Pg 168-170  AH-Chap 3:  Pg 182  AA-Chap 4:  Pg 174 &175 |  |
| **10** | 19-04-2021 to  23-04-2021 | Left recursion , left factoring  Simple and fast implementation of predictive parsers  Non recursive Predictive Parsing | AA-Chap 4:  Pg 176-187 | **Quiz#2** |
| **11** | 26-04-2021 to  30-04-2021 | First and Follow sets  Mapping of parsing table with Stack implementation  Predictive parser with examples LL(1) Grammar | AA-Chap 4:  Pg 188-190,192  AH-Chap 4:  Pg 213-217 |  |
| **12** | 03-05-2021 to  07-05-2021 | Bottom up Parsing Technique  Handles, handle pruning and stack implementation of SLR  Shift-reduce parsers examples | AA-Chap 4:  Pg 195-199  AH-Chap 5:  Pg 338  AA-Chap 4:  Pg 218-220 |  |
| **13** | 10-05-2021 to  14-05-2021 | Canonical collection  parsing table and moves made by the LR parser  parsing table and LR parser examples | AA-Chap 4:  Pg 220-225 | Assignment # 3 |
| **14** | 17-05-2021 to  21-05-2021 | Over view of Syntax Directed Translation, Semantic Rules ,Inheritance, synthesized attributes  Inherited & synthesized attributes examples L& S attributed Grammar  Construction of Syntax Tree & Dependency graph | AA-Chap 5:  Pg 280-297 | **Quiz#3** |
| **15** | 24-05-2021 to  28-05-2021 | Translation Scheme  Over view of Type checking , Type systems ,type checking of statements  Type checking of Expressions ,Type checking of functions | AA-Chap 5:  Pg 297-300  AA-Chap 6:  Pg 300- 344 |  |
| **16** | 31-05-2021 to  04-06-2021 | Example of statements and expression type checking  Revision of Topics before Final Exams  Paper Discussion and students query Session. | AA-Chap 6:  Pg 350-352 |  |
| **Final Examination**  **(07-06-2021 to 19-06-2021)** | | | | |

**LAB PLAN**

**Course Title:**

**Course Code:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Week No.** | **Lab Date** | **Objective** | **Required Reading** |
| **1** | 15-02-2021 to  19-02-2021 | Overview of Compilers with e.g and explore the IDEs by Making simple program which recognizes the key strokes as you press different letters |  |
| **2** | 22-02-2021 to  26-02-2021 | Make a program which is capable of Saving & opening a given text in a file.  Searching for a given string in a file, Replace the searched string with the given string inorder to give the sense of identifying the lexeme and generates the Token. |  |
| **3** | 01-03-2021 to  05-03-2021 | Write a lexical analyzer in any language for the given tokens |  |
| **4** | 08-03-2021 to  12-03-2021 | This lab covers some efficiency issues concerned with the buffering of input. First we’ll look at the two-buffer input scheme that is useful when look-ahead on the input is necessary to identify the tokens. |  |
| **5** | 15-03-2021 to  19-03-2021 | This lab, we’ll explore a useful technique for speeding up the lexical analyzer, i.e. the use of ***sentinels*** called **Buffering Technique II** |  |
| **6** | 22-03-2021 to  26-03-2021 | Consider the following regular expression:  **(a | b) (ba | ab)\***  Construct the transition diagram for the above regular expression and implement it in any conventional programming language. |  |
| **7** | 29-03-2021 to  02-04-2021 | This lab is designed in order that how the tokens and Identifiers get recognition in a given input string. |  |
| **8** | **Mid Term Examination**  **(05-04-2021 to 10-04-2021)** | | |
| **9** | 12-04-2021 to  16-04-2021 | Designing of context-free grammar for your own language. It must cover Multiple declaration, Loops, conditional statements User define functions. |  |
| **10** | 19-04-2021 to  23-04-2021 | Implementation of Lexical Analyzer in any programming language for **your context-free grammar.**  The steps are:   1. Identification of Tokens 2. Specification of Tokens in terms of Regular Expressions 3. Recognition of Tokens in terms of Transition diagrams 4. Coding of Lexical analyzer |  |
| **11** | 26-04-2021 to  30-04-2021 | Transform your Grammar in the unambiguous form and perform the left recursion and left factoring from the grammar. |  |
| **12** | 03-05-2021 to  07-05-2021 | Find out the First () and Follow () of the grammar which you have completed in the previous lab. |  |
| **13** | 10-05-2021 to  14-05-2021 | Implement the Non recursive predictive parser by mapping the functions of First () & Follow () and making the Parsing table, perform the stack implementation by taking any program of your own language and shows its output. |  |
| **14** | 17-05-2021 to  21-05-2021 | Transform Your Grammar in Augmented form in order to Implement SLR parser and find the closure and Goto operation. |  |
| **15** | 24-05-2021 to  28-05-2021 | Write the complete code of SLR parser with the help of making Transition diagram, also show its ouput. Take any program of your own language as an input and display the output. |  |
| **16** | **Lab Examination**  **(31-05-2021 to 04-06-2021)** | | |