**BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI**

**DEPARTMENT OF COMPUTER SCIENCE AND INFORMATION SYSTEMS**

**Compiler Construction (CS F363)** 

**II Semester 2024-25**

**Compiler Project**

**Coding Details**

**(March 15, 2025)**

1. Team Members Names and IDs

ID **2022A7PS0160P** Name: **Aditya Agrawal**

ID **2022A7PS1186P** Name: **Shashank Jha**

ID **2022A7PS0096P** Name: **Samarth**

ID **2022A7PS1170P**  Name: **Shikhar Singh**

1. Mention the names of the Submitted files :

1: parserDef.h 7: associativeArray.h 13: coding details.docx

2: parser.h 8: associativeArray.c 14\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

3: parser.c 9: set.h 15\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

4: lexerDef.c 10: set.c 16\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

5: lexer.h 11: makefile 17\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

6: lexer.c 12: driver.c 18\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Total number of submitted files (including copy the pdf file of this coding details pro format) : 13 (All files should be in ONE folder named as Group\_#)
2. Have you compressed the folder as specified in the submission guidelines? (yes/no) **YES**
3. **Lexer Details:**
   1. Technique used for pattern matching: **Deterministic Finite Automata(DFA)**
   2. Keyword Handling Technique:**Lookup Table**
   3. Hash function description, if used for keyword handling:

**hash(str)=str[0]\*31+str[1]\*31²+str[2]\*31³+...+str[n-1]\*31ⁿ**

All operations performed using **modulo arithmetic**, with modulus = max lookup table capacity (250 in our case)

* 1. Have you used twin buffer? (yes/ no):  **YES**
  2. Error handling and reporting (yes/No): **YES**
  3. Describe the errors handled by you : **Unknown symbols, unknown patterns, length of identifiers exceeding 20 characters**
  4. Data Structure Description for tokenInfo (in maximum two lines): **Line number of token (int), type of token (char \*), lexeme (char \*), identifier for the last token of the input file (bool)**

1. **Parser Details:** 
   1. High Level Data Structure Description (in maximum three lines each, avoid giving C definitions used):
      1. grammar : arrays of strings to store nonterminals, terminals. array of ProductionRules, which have an lhs string and an array of rhs string to store rules. readGrammar Function used to read grammar from file grammar.txt.
      2. FIRST and FOLLOW sets: two arrays of Sets for first and follow info for each non terminal. Sets implemented by array.
      3. parse table: Associative Array used to go from a pair of strings to Production rule
      4. parse tree: (Describe the node structure also): ParseTreeNodes are storing the symbols and associated tokenInfo, along with a boolean telling if it’s terminal. ParseTree just stores the root node.
      5. Any other (specify and describe)

Set: implemented by array. Union, inclusion implemented

Associative Array: Polynomial String hashing used. Implemented by array of linked lists. Insertion, lookup implemented

* 1. Parse tree
     1. Constructed (yes/no): **YES**
     2. Printing as per the given format (yes/no): **YES**
     3. Describe the order you have adopted for printing the parse tree nodes (in maximum two lines): IN-order Traversal (DFS with parent nodes followed by DFS on each child)
  2. Grammar and Computation of First and Follow Sets
     1. Data structure for original grammar rules: **ProductionRule**
     2. FIRST and FOLLOW sets computation automated (yes /no): **NO**
     3. Name the functions (if automated) for computation of First and Follow sets: **N.A**
     4. If computed First and Follow sets manually and represented iin file/function (name that) : computeFirstAndFollowSets() in parser.c
  3. Error Handling
     1. Attempted (yes/ no): **YES**
     2. Describe the types of errors handled: SYNSET/ERR handled

1. Compilation Details:
   1. Makefile works (yes/no):**YES**
   2. Code Compiles (yes/ no):**YES**
   3. Mention the .c files that do not compile: **NA**
   4. Any specific function that does not compile: **NA**
   5. Ensured the compatibility of your code with the specified gcc version (yes/no) **YES**
2. Driver Details: Does it take care of the options specified earlier(yes/no): **YES**
3. Execution
   1. status (describe in maximum 2 lines): Lexical analyzer fully functional. Some false errors reported during execution of Syntax Analyzer
   2. Gives segmentation fault with any of the test cases (1-6) uploaded on the course page. If yes, specify the testcase file name: **NO**
4. Specify the language features your lexer or parser is not able to handle (in maximum one line) **NA**
5. Are you availing the lifeline (Yes/No): **NO**
6. Declaration: We, **Aditya Agrawal, Samarth, Shashank Jha, Shikhar Singh** declare that we have put our genuine efforts in creating the compiler project code and have submitted the code developed only by us. We have not copied any piece of code from any source. If our code is found plagiarized in any form or degree, we understand that a disciplinary action as per the institute rules will be taken against all of us in our team and we will accept the penalty as decided by the department of Computer Science and Information Systems, BITS, Pilani.

Your names and IDs

Name: Aditya Agrawal ID: 2022A7PS0160P

Name: Shikhar Singh ID: 2022A7PS1170P

Name: Shashank Jha ID: 2022A7PS1186P

Name: Samarth ID: 2022A7PS0096P

Date: March 15th, 2025

-------------------------------------------------------------------------------------------------------------------------------------------------

*Not to exceed 3 pages.*