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)

Group Number
1

1. Team Members Names and IDs

ID 2022A7PS0074P	Name Saaransh Jain
ID 2022A7PS0152P	Name Aman Patel
ID 2022A7PS0094P	Name Vishnu Hari
ID 2022A7PS0177P	Name Parth Sudan
ID ______________________	Name______________________
ID ______________________	Name______________________

2. Mention the names of the Submitted files :

1 driver.c	7 lexer.h	13 parser.c	19 testcase4.txt
2 firstAndFollow.c	8 lexerDef.h	14 parser.h	20 testcase5.txt
3 firstAndFollow.h	9 lookup.c	15 parserDef.h	21 testcase6.txt
4 grammar.h	10 lookup.h	16 testcase1.txt	22 coding details.pdf
5 grammar.txt	11 makefile	17 testcase2.txt	
6 lexer.c	12 parseGrammar.py	18 testcase3.txt	

3. Total number of submitted files (including copy the pdf file of this coding details pro forma) : ____22_____ (All files should be in ONE folder named as Group_#)

4. Have you compressed the folder as specified in the submission guidelines? (yes/no)_____yes_______

5. **Lexer Details:**
   [A]. Technique used for pattern matching: A DFA was constructed on paper and was implemented using techniques similar to how Moore machines are written in code; a while loop to break at end of file and a switch case statement to transition between states. To recognise conditions, simply the ASCII value of characters was used.
   [B]. Keyword Handling Technique: Keywords are prefilled into the lookup table and any time a pattern may clash with a keyword, the lookup table is queried for the lexeme. If found, the prefilled entry is returned instead.
   [C]. Hash function description, if used for keyword handling: A variation of polynomial rolling hash function, where each character is multiplied by an exponentially growing factor and added. Growth of factor is controlled by mod 0x7fff so that it doesn't grow too large.
   [D]. Have you used twin buffer? (yes/ no) yes
   [E]. Error handling and reporting (yes/No): yes
   [F]. Describe the errors handled by you: Unrecognised patterns, Unexpected symbols in the middle of patterns, variable identifier length check and function identifier length check are handled.
   [G]. Data Structure Description for tokenInfo (in maximum two lines): A structure that holds the tokentype as an enum(internally an integer), the lexeme string and length and numeric converted lexemes.

6. **Parser Details:**

[A]. High Level Data Structure Description (in maximum three lines each, avoid giving C definitions used):

i. grammar: A grammar.txt was created that holds the grammar in human readable format. We wrote a python script to convert this txt to a c header that creates the grammar in memory. Grammar has its start symbol and rules. Each rule has an lhs, a bool flag for epsilon and an rhs which is an array of symbols (terminal or otherwise).

ii. FIRST and FOLLOW sets: Implemented as an array indexable by the non terminal. Each element of the array has two attributes - firstSet and followSet. Each of these is an unsigned 64 bit bitmask (integer), which is sufficient to represent subsets of a set of size 59 (number of terminals).

iii. parse table: ParseTable is simply an array of Rules (NULL if no transition exists). It is indexable by the non terminal symbol at the top of the stack and the terminal symbol at the lookahead of the input stream.

iv. parse tree: (Describe the node structure also): Parse Tree simply has a reference to its root. Each node has a marker for whether it is a terminal node or not, along with the non terminal if it isn't, or the token, lexeme (string/int/float) if it is. Also contains the line number and references to its children and parent.

v. Any other (specify and describe) The stack is implemented as an array of TreeNodes. Starting capacity of the stack is 32 nodes, capacity doubles whenever more space is required, in accordance with standard practices.

[B]. Parse tree

i. Constructed (yes/no): yes

ii. Printing as per the given format (yes/no): yes

iii. Describe the order you have adopted for printing the parse tree nodes (in maximum two lines) inorder has been used. For more than 2 children, the inorder specification used is firstChild, parent, restChildren.

[C]. Grammar and Computation of First and Follow Sets

i. Data structure for original grammar rules: Created automatically from grammar.txt by a python script. The python script is not used anywhere else or during compilation at all.

ii. FIRST and FOLLOW sets computation automated (yes /no) yes

iii. Name the functions (if automated) for computation of First and Follow sets: computeFirstSets and computeFollowSets, which call firstSetOfRule and followSetOfNT (helper functions) respectively.

iv. If computed First and Follow sets manually and represented in file/function (name that): N/A

[D]. Error Handling

v. Attempted (yes/ no): yes

vi. Describe the types of errors handled: When the token at the top of the stack does not match the token at the lookahead of the token stream, when there is no entry in the parse table to go from the non terminal at the top of the stack to the token at the lookahead of the token stream and if the token stream ends before the stack empties.

7. Compilation Details:

[A]. Makefile works (yes/no): yes

[B]. Code Compiles (yes/ no): yes

[C]. Mention the .c files that do not compile: N/A

[D]. Any specific function that does not compile: N/A

[E]. Ensured the compatibility of your code with the specified gcc version (yes/no) yes

8. Driver Details: Does it take care of the options specified earlier(yes/no): yes

9. Execution

[A]. status (describe in maximum 2 lines): The compiler is capable of removing comments, generating the token stream and the parse tree, all while detecting and recovering from both lexical and syntax errors.

[B]. Gives segmentation fault with any of the test cases (1-6) uploaded on the course page. If yes, specify the testcase file name: No segmentation fault with any of the test cases (1-6)

10. Specify the language features your lexer or parser is not able to handle (in maximum one line): N/A

11. Are you availing the lifeline (Yes/No): No

12. Declaration: We, Saaransh Jain, Aman Patel, Vishnu Hari and Parth Sudan 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: Saaransh Jain	ID: 2022A7PS0074P
Name: Aman Patel	ID: 2022A7PS0152P
Name: Vishnu Hari	ID: 2022A7PS0094P
Name: Parth Sudan	ID: 2022A7PS0177P
Name:______________________	ID: _______________________________
Name:______________________	ID: _______________________________

Date: _______________

Not to exceed 3 pages.