BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI
DEPARTMENT OF COMPUTER SCIENCE AND INFORMATION SYSTEMS

Compiler Construction (CS F363)
II Semester 2023-24
Compiler Project
Coding Details
(March 5, 2024)

Group Number

4

---

1. Team Members Names and IDs

ID : 2020B3A70906P	Name : Rohit Raj
ID : 2020B5A70924P	Name : Adarsh Agarwal
ID : 2020B5A70693P	Name : Nandalal Odedara
ID : 2020B3A70917P	Name : Gaurav Mishra
ID : 2020B2A71611P	Name : Nikhil Agarwal
ID : 2020B3A70752P	Name : Parivesh Bajpai

2. Mention the names of the Submitted files:
    1. grammar.txt
    2. coding details.pdf
    3. lexerDef.h
    4. lexer.h
    5. lexer.c
    6. stack.h
    7. stack.c
    8. hashtable.h
    9. hashtable.c
    10. parserDef.h
    11. parser.h
    12. parser.c
    13. driver.c
    14. makefile
    15. t1.txt
    16. t2.txt
    17. t3.txt
    18. t4.txt
    19. t5.txt
    20. t6.txt
    21. testcase1.txt
    22. testcase2.txt
    23. testcase3.txt

3. Total number of submitted files (including copy of the pdf file of this coding details pro forma): 23

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

5. **Lexer Details:**
    [A]. Technique used for pattern matching: Case-based switch statements and derivation of the cases from the underlying DFA. If the length of the input is n, the time taken is O(n). If a comment is encountered it is not tokenised. If error is recognised as per the DFA, the relevant output is printed.
    [B]. Keyword Handling Technique: Hash function is generated using chaining.
    [C]. Hash function description, if used for keyword handling: Iteration through each input character of the input key and updation of the hash value by left-shifting it by 5 and adding the ASCII value of the current character.
    [D]. Have you used twin buffer? (yes/ no) YES
    [E]. Error handling and reporting (yes/No): YES
    [F]. Describe the errors handled by you : If the input starts with an unknown symbol, an error is reported directly. If there is an error as per the DFA, the error is printed with the appropriate line number , the lexeme value, which is invalid. In the case where the length of function identifier is greater than 30 and length of variable identifier is greater than 20, appropriate message with a line number is quoted.

    [G]. Data Structure Description for tokenInfo (in maximum two lines): The data structure tokenInfo contains three variables i.e., value which is a string contains the lexeme value, tkId which is a string containing the token name and linenumber of integer type which shows line number of the token in the code.

6. **Parser Details:**

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

i. grammar: The structure consists of information about the total elements in a particular grammar rule i.e. left elements and right elements, the left element, and an array of right elements. It also has a rule number that is associated with the line number in the grammar.txt file.

ii. FIRST and FOLLOW sets : It contains a non-terminal array of first and follow sets, along with a number of elements in its first and follow sets. It also takes care if any non-terminal derives an epsilon. It further also contains bools - namely *visited(Purpose: If the first has already been computed, it will use it i.e. memoization for time optimisation)*, *followCalc(Purpose: to check if follow has been pre computed for that particular non-terminal)* and *lock(used in follow algorithm to detect loop)* which are used for intermediate steps.

iii. Parse Table: Parse Table is a 2-D array with rows for non-terminals (unique) and columns for number of terminals including '$' symbol indicating end of file. The entry for a particular row and column has value as -1 for error, -2 for synchronization set and value greater than equal to 0 indicates the corresponding grammar rule number.

iv. parse tree: (Describe the node structure also) : Each Node of the tree has 6 variables i.e., lexeme which contains the value of the node, children which in a pointer of arrays to the children, number of children variable, the parent node identifier varible(parentIndex) and the unique identifier of itself(nodeIndex).

v. Any other (specify and describe): The stack element has been modified in such a way that it point to the address in the tree so that the line number and other details can be accessed and modified using the stack element.There is a look-up table (NTLookupEntry) made for tracking the grammar rule number for the non-terminal , non terminal and the index in the first follow data structure for that specific non-terminal. The errors are printed line number wise in a single traversal while parsing the tokens. There is **no** extra data structure used for handling errors.

[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 Traversal - Inorder traversal means traversing a tree in a format that is the left chid first then the root node and then the right child.

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

i. Data structure for original grammar rules: 1D array of struct Grammar rule which contains one 1 D array of right elements of a grammar rule, one string variable containing the left rule, and one integer containing the number of right elements in a rule.
ii. FIRST and FOLLOW sets computation automated (yes /no): Yes
iii. Name the functions (if automated) for computation of First and Follow sets: a) **initialiseFFandLookUp()** : to initialize the first follow table and lookup with the non-terminals in the grammar.
computeFirstandFollowSets(): This function implicitly calls populateFirst() and populateFollow() in respective order for each non terminals. Follow set is computed after the first set of all non terminals are computed.
Other functions are used for filling the sets of first and follow uniquely.
iv. If computed First and Follow sets manually and represented in file/function (name that) : NA

[D]. Error Handling

v. Attempted (yes/ no): Yes
vi. Describe the types of errors handled:
Lexical errors - Unknown pattern or symbol, identifiers exceeding maximum specified length
Syntax error - Illegally positioned or missing tokens

7. Compilation Details:
    [A]. Makefile works (yes/no): yes
    [B]. Code Compiles (yes/ no): yes
    [C]. Mention the .c files that do not compile: NA
    [D]. Any specific function that does not compile: NA
    [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 code executes correctly for all the given test cases(t1-t6) and as well for self made test cases(testcase1 is a syntactically correct long testcase, testcase2 has >1000 lines of code and testcase3 is a long testcase with errors both lexical and syntax). There is a function for printing the first and follow sets that can be called to check the correctness of automation.
    [B]. Gives segmentation fault with any of the test cases (1-6) uploaded on the course page. If yes, specify the test case file name: None

10. Specify the language features your lexer or parser is not able to handle (in maximum one line) : None

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

12. Declaration: We, Rohit Raj, Adarsh Agarwal, Nikhil Agarwal, Gaurav Mishra, Nandalal Odedara, Parivesh Bajpai 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 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 : Rohit Raj	ID : 2020B3A70906P
Name : Adarsh Agarwal	ID : 2020B5A70924P
Name : Nandalal Odedara	ID : 2020B5A70693P
Name : Gaurav Mishra	ID : 2020B3A70917P
Name : Nikhil Agarwal	ID : 2020B2A71611P
Name : Parivesh Bajpai	ID : 2020B3A70752P

Date: 5 March 2024
