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

Group No. 21

Compiler Construction (CS F363)
II Semester 2019-20
Compiler Project (Stage-1 Submission)
Coding Details
(February 24, 2020)

---

1. IDs and Names of team members

ID: 2017A7PS1195P Name: Anirudh S Chakravarthy

ID: 2017A7PS0134P Name: S Hariharan

ID: 2016B2A70770P Name: Honnesh Rohmetra

ID:2017A7PS1174P Name: Praveen Ravirathinam

2. Mention the names of the Submitted files :

1. driver.c	7. lexertester.c	13. parsertester.c
2. lexer.c	8. parser.c	14. makefile
3. lexer.h	9. parser.h	15. grammar.txt
4. hash.c	10. parserutils.c	16. parsetable.txt
5. hash.h	11. parserutils.h	17. README.md
6. lexerDef.h	12. parserDef.h	

3. Total number of submitted files: 24 (including testcases and makefile)

4. Have you mentioned your names and IDs at the top of each file (and commented well)? (Yes/ no) Yes
[Note: Files without names will not be evaluated]

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

6. Lexer Details:

[A]. Technique used for pattern matching: Lookahead enabled DFA

[B]. DFA implementation (State transition using switch case, graph, transition table, any other (specify): state transitions implemented using switch cases.

[C]. Keyword Handling Technique: Hashing

[D]. Hash function description, if used for keyword handling: Sum of squares of ASCII values of characters.

[E]. Have you used twin buffer? (yes/ no): Yes

[F]. Lexical error handling and reporting (yes/No): Yes

[G]. Describe the lexical errors handled by you: identifier length > 20, invalid character, reading only a single '.' or '=', RNUM like 12.<others> and 12.03e<others>.

[H]. Data Structure Description for tokenInfo (in maximum two lines): Tokenid mapping to enum of terminals, lexeme, and line number of the token (called token in our implementation).

[I]. Interface with parser: Parser utilizes getNextToken() from lexer to increment lookahead pointer.

7. Parser Details:

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

i. grammar: Array of rules of size *_NUM_RULES*. Each rule is a structure containing the LHS of the rule (nonterminal) and a linked list containing all RHS symbols (rhsnode structure) of the rule.

ii. parse table: a 2D integer array of size (*_NUM_NONTERMINALS*) x (*_NUM_TERMINALS*). An entry in the parse table corresponds to the rule number to be used for parsing on encountering that terminal, for a given stack configuration (with a nonterminal at the top). Error and Syn entries are also included appropriately.

iii. parse tree: a structure containing a flag (leaf node or not), and pointers to the node's parent, right sibling, and leftmost child. Data is encapsulated in a union, corresponding to leaf node (token) or non-leaf node (nonterminal).

iv. Parsing Stack node structure: contains a symbol (a union of terminal or non-terminal), a tag to differentiate between the two, a pointer to the next element in the stack, and a pointer to the corresponding node in the parse tree.

v. Any other (specify and describe): NA

[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): In-order traversal of the n-ary parse tree. Traverse the leftmost child, followed by the node's data, and subsequently the other children of the node.

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

i. Data structure for original grammar rules: structure(rhsnode) containing a symbol (union of terminal or nonterminal), a tag for differentiating. It is doubly linked (previous and next pointers) for efficient traversal of the symbols in the RHS of the rule.

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

iii. Data structure for representing sets: Bitset (unsigned long long int). Bit[i] = 1 implies terminal indexed by enumeration i is present in the set.

iv. Time complexity of computing FIRST sets: O(num_terminals * num_nonterminals) in worst case.

v. Name the functions (if automated) for computation of First and Follow sets: computeFirstSets(), computeFollowSets(), and computeFirstAndFollowSets().

vi. If computed First and Follow sets manually and represented in file/function (name that): NA

[D]. **Error Handling**

i. Attempted (yes/ no): Yes

ii. Printing errors (All errors/ one at a time): One at a time

iii. Describe the types of errors handled: If terminal on stack doesn't match lookahead token, if lookahead token doesn't belong to first set of nonterminal on the top of the stack.

iv. Synchronizing tokens for error recovery (describe): If top of the stack is a terminal, discard tokens until the same terminal is read. If it is a nonterminal, discard tokens until a token belonging in the follow set of the nonterminal is encountered.

v. Total number of errors detected in the given testcase t6(with_syntax_errors).txt: 18 total – 2 lexical and 16 syntactical. Out of the 16, 10 are syntax error and 6 correspond to syn.

8. 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

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

10. Execution

[A]. status (describe in maximum 2 lines): Created stage1exe as required using the makefile. Successfully executes and terminates on given test cases.

[B]. Execution time taken for

- t1.txt (in ticks) 297 and (in seconds) 0.000297 (refer lexertester.c for timing)
- t2.txt (in ticks) 527 and (in seconds) 0.000527
- t3.txt (in ticks) 590 and (in seconds) 0.000590
- t4.txt (in ticks) 2071 and (in seconds) 0.002071
- t5.txt (in ticks) 2146 and (in seconds) 0.002146
- t6.txt (in ticks) 1504 and (in seconds) 0.001504

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

11. Specify the language features your lexer or parser is not able to handle (in maximum one line): None that we are aware of.

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

13. Declaration: We, Anirudh S Chakravarthy, S Hariharan, Honnesh Rohmetra and Praveen Ravirathinam, declare that we have put our genuine efforts in creating the compiler project code and have submitted the code developed only by our group. 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 us and we will accept the penalty as decided by the department of Computer Science and Information Systems, BITS, Pilani. [Write your ID and names below]

ID 2017A7PS1195P	Name: Anirudh S Chakravarthy
ID 2017A7PS0134P	Name: S Hariharan
ID 2016B2A70770P	Name: Honnesh Rohmetra
ID 2017A7PS1174P	Name: Praveen Ravirathinam

Date: 24/02/2020
