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

Group No. 2

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:	2017A7PS0116P	NAME:	AYUSH SINGHAL
ID:	2017A7PS0001P	NAME:	BHOOMI SAWANT
ID:	2017A7PS0086P	NAME:	PRATIK R KAKADE
ID:	2017A7PS0218P	NAME:	SAKSHAM GUPTA
ID:	2017A7PS0114P	NAME:	YASHDEEP GUPTA

2. Mention the names of the Submitted files :

1 driver.c
2 grammar.txt
3 grammar_Init.c
4 grammar_Init.h
5 grammar_InitDef.h
6 lexer.c
7 lexer.h
8 lexerdef.h
9 makefile
10 parser.c
11 parser.h
12 parserdef.h
13 printParseTree.c
14 stack.c
15 stack.h
16 stackDef.h
17-22 t1.txt - t6.txt

3. Total number of submitted files: 22 (All files should be in **ONE folder** named exactly as Group_#, # is your group number)

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:

A DFA implementation for regular expressions was used.

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

[C]. Keyword Handling Technique: Once an identifier is found, an entry is checked in the lookup Table.

[D]. Hash function description, if used for keyword handling: NA

[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:

1) If we encounter a character for which no state transition is defined, a Lexical error is raised.

2) If the length of an identifier exceeds 20, a lexical error is raised.

[H]. Data Structure Description for tokenInfo (in maximum two lines):

A character array for lexeme, unsigned int for line number, an enum for tokens and a tagged union structure for value(NUM and RNUM) has been implemented.

[I]. Interface with parser

To initialise the lexer, initLexer(inFile) is called by the parser. Further, the getNextToken() function can be called by the parser to retrieve the next valid token from the lexer.

7. Parser Details:

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

i. Grammar :

An array of structure “cell” has been used to store each grammar rule. It stores the Enum value of the LHS Non terminal. It also stores the pointer to the first token in its RHS which is stored as a linked list, each containing enum value, tag and pointer to the next node.

ii. Parse table

Parsing table is implemented as a 2-d matrix of integers with Non terminal enum being used as index for row and Terminal enum for column. Each cell stores the rule number corresponding to the transition or -1 in case of error or -2 for syn.

iii. parse tree: (Describe the node structure also)

A tagged union to store the symbol, left child and right sibling pointers, a token to store the info and an errorFlag to indicate errors have been used.

iv. Parsing Stack node structure :

A stack is an array of stackItems with a top pointer and a capacity. Each stackItem contains a tagged symbol and a pointer to the corresponding parse tree node.

v. Any other (specify and describe)

grammarInit.c stores the defintions of initGrammar() to initialise everything from the grammar file. First and follow sets computations have also been done in this file.

[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 has been used for printing the nodes.

The format is : leftMostChild → Parent Node → Right siblings

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

i. Data structure for original grammar rules

An array of structure “cell” has been used to store each grammar rule. It stores the Enum value of the LHS Non terminal. It also stores the pointer to the first token in its RHS which is stored as a linked list, each containing enum value, tag and pointer to the next node.

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

iii. Data structure for representing sets

An array of 3 integers have been used for representing first and follow sets. Each bit position corresponds to the enum value of a Terminal.

iv. Time complexity of computing FIRST sets

O(TotalRules * MaxNoOfTokensInRule * K)

Here K is the number of iterations for first set to converge

v. Name the functions (if automated) for computation of First and Follow sets

computeFirstSet()computeFollowSet() computeFirstAndFollowSets()

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

[D]. **Error Handling**

i. Attempted (yes/ no): YES

ii. Printing errors (All errors/ one at a time) : All errors printed line by line as they are encountered

iii. Describe the types of errors handled

Lexical Errors of any kind as well as Syntax errors have been handled.

iv. Synchronizing tokens for error recovery (describe)

The tokens in the follow set of the nonTerminal on the top of stack have been used for synchronisation.

v. Total number of errors detected in the given testcase t6(with_syntax_errors).txt

19 syntax errors and 2 lexical errors detected on 11 unique lines.

8. Compilation Details:

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

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

[C]. Mention the .c files that do not compile:NONE

[D]. Any specific function that does not compile:NONE

[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):Executes correctly on all given test cases.

[B]. Execution time taken for

- t1.txt (in ticks) 3080 and (in seconds) 0.003080
- t2.txt (in ticks) 927 and (in seconds) 0.000927
- t3.txt (in ticks) 671 and (in seconds) 0.000671
- t4.txt (in ticks) 1341 and (in seconds) 0.001341
- t5.txt (in ticks) 1344 and (in seconds) 0.001344
- t6.txt (in ticks) 1938 and (in seconds) 0.001938

[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) CommentMarks with no ending(** at end) are not tokenised as comments.

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

13. Declaration: We, AYUSH SINGHAL, BHOOMI SAWANT, PRATIK R KAKADE, SAKSHAM GUPTA, YASHDEEP GUPTA (your names) 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.

ID: 2017A7PS0116P NAME: AYUSH SINGHAL

ID: 2017A7PS0001P NAME: BHOOMI SAWANT

ID: 2017A7PS0086P NAME: PRATIK R KAKADE

ID: 2017A7PS0218P NAME: SAKSHAM GUPTA

ID: 2017A7PS0114P NAME: YASHDEEP GUPTA

Date: 24/02/2020