BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI
DEPARTMENT OF COMPUTER SCIENCE AND INFORMATION SYSTEMS
Compiler Construction (CS F363)
II Semester 2019-20
Compiler Project (Stage-1 Submission)
Coding Details
(February 24, 2020)

Group No. 33

---

1. IDs and Names of team members

ID: **2017A7PS0023P** Name: **AKSHIT KHANNA**

ID: **2017A7PS0030P** Name: **SWADESH VAIBHAV**

ID: **2017A7PS0077P** Name: **ARYAN MEHRA**

ID: **2017A7PS0429P** Name: **VIPIN BASWAN**

2. Mention the names of the Submitted files :

1 **driver.c**
2 **lexer.h**
3 **lexerDef.h**
4 **lexer.c**
5 **parser.h**
6 **parserDef.h**
7 **parser.c**
8 **makefile**
9 **grammar.txt**
10 **t1.txt**
11 **t2.txt**
12 **t3.txt**
13 **t4.txt**
14 **t5.txt**
15 **t6.txt**
16 **Coding Details Stage1**

3. Total number of submitted files: **16** (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: **We have used DFA based implementation of lexical analyzer using a switch case construct for efficient and intuitive state transitions. We use 1 look ahead only because our language is LL(1).**

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

[C]. Keyword Handling Technique: **The keywords are checked through hash table ADT implementation**

[D]. Hash function description, if used for keyword handling: **We use the prime value 31 to convert the string to a 31-ary number (thus incorporating both value and positional variance) and then a modulo is taken with the hash table size. Conflicts are handled through chained linked lists**.

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

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

[G]. Describe the lexical errors handled by you: **All lexical errors are handled in acc. with specifications.**

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

Token structure consists of 4 fields : (char * token , char * lexeme , void * value , int lineNum)

[I]. Interface with parser: **Lexical analyzer is an obedient servant of the parser. The lexical analyzer calls the getnexttoken() which uses the nexttoken() that is implemented as a DFA switch case simulator**.

7. Parser Details:

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

i. grammar: **It is an array of linked lists of nodes. Each linked list is analogous to a rule of the grammar. The head of the list acts as the LHS Non Terminal followed by the RHS. Each Node is a union of Non-terminal or Terminal structures.**

ii. parse table: **It is a 2D array with rows being referenced with the enumerated Non-Terminals and columns by the Terminals (including EOF/$). The value is the rule number that is applicable, with -1 representing error and -2 representing SYN set according to follow of the Non-Terminal.**

iii. parse tree: (Describe the node structure also): **Tree Node consists of a union of leaf or non leaf structure, along with one right side sibling and leftmost child pointer each. Leaf consists of a token and non leaf has a Non-Terminal.**

iv. Parsing Stack node structure : **Stack Node has a next pointer to the node below it. It itself contains the pointer to the TreeNode that it refers to, hence saving memory and making error recovery easy.**

v. Any other (specify and describe): **Keeping with the spirit of compilation and not stopping on encountering an error, we maintain an Error LinkedList(s) of Lexical and Syntactical errors. The linkedlist is separate but the Error node structure is common with a union over lexical and syntactical error structures.**

[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 with Leftmost child -> Parent -> Other siblings of the child in order

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

i. Data structure for original grammar rules: **Array of Linked Lists with Node being union of Terminal and Non-Terminal.**

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

iii. Data structure for representing sets: **Integer arrays are used for this purpose.**

iv. Time complexity of computing FIRST sets: **We are using the best possible DFS like approach with dynamic programming 2D array to recursively calculate the first set. The Non terminals act as vertices (visited, not visited, visited but in progress) So the theoretical net worst case time complexity is O(R*M*T) where R is the number of rules, M is the maximum Non Terminals on the right side of a rule that can have EPSILON in their**

first set and T is the number of Terminals. In general (practically) the function performs much better because M<=3 or 4.

v. Name the functions (if automated) for computation of First and Follow sets: **calculateFirstSet() and calculateFollowSet() with utilities initialiseFirstSet() and initialiseFollowSet()**

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 at once**

iii. Describe the types of errors handled: **Lexical and Syntax Errors (both) are captured in seperate lists. They use union within them. The lists keep building on the fly and are displayed at the end.**

iv. Synchronizing tokens for error recovery (describe): **We use the classical SYN set of panic mode recovery where the SYN set of a non terminal has elements from its follow set which are not corresponding to any rules beforehand.**

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

12 Syntactic Errors and 2 Lexical Errors

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): **Fully executing code, all errors being caught and handled, no segmentation faults,**

[B]. Execution time taken for

- t1.txt (in ticks) **846.00** and (in seconds) **0.000846**
- t2.txt (in ticks) **900.00** and (in seconds) **0.000900**
- t3.txt (in ticks) **1804.00** and (in seconds) **0.001804**
- t4.txt (in ticks) **1568.00** and (in seconds) **0.001568**
- t5.txt (in ticks) **2439.00** and (in seconds) **0.002439**
- t6.txt (in ticks) **1266.00** and (in seconds) **0.001266**

[C]. 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**

11. Specify the language features your lexer or parser is not able to handle (in maximum one line) : **None. Even out of alphabet errors like usage of % or $ characters are handled very well.**

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

13. Declaration: We, **Akshit Khanna, Swadesh Vaibhav, Aryan Mehra and Vipin Baswan** (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. [Write your ID and names below]

ID: **2017A7PS0023P** Name: **AKSHIT KHANNA**

ID: **2017A7PS0030P** Name: **SWADESH VAIBHAV**

ID: **2017A7PS0077P** Name: **ARYAN MEHRA**

ID: **2017A7PS0429P** Name: **VIPIN BASWAN**

Date: **24/02/2020**

Should not exceed 4 pages.