•

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)

---

1. IDs and Names of team members

ID:2016B4A70614P	Name: Anirudh Buvanesh
ID:2016B4A70824P	Name: Anubhav Bansal
ID:2016B5A70716P	Name: Aayush Agarwal
ID:2016B5A70607P	Name: Manan Agarwal
ID:2016B4A70636P	Name: Ashish Kumar

2. Mention the names of the Submitted files :

1	lexer.c	7	driver.c
2	parser.c	8	grammar.txt
3	lexer.h	9	makefile
4	lexerDef.h		
5	parser.h		
6	parserDef.h		

3. Total number of submitted files: 9

4. Have you mentioned your names and IDs at the top of each file (and commented well)? Yes

5. Have you compressed the folder as specified in the submission guidelines? Yes

6. Lexer Details:

[A]. Technique used for pattern matching: Maximal Munch Principle

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

Switch Case.

[C]. Keyword Handling Technique: Hash Table

[D]. Hash function description, if used for keyword handling: Polynomial rolling hash function.

[E]. Have you used twin buffer? : Yes

[F]. Lexical error handling and reporting : Yes

[G]. Describe the lexical errors handled by you

lexeme length >20,rnum values not written properly, operators, identifier of all format.

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

struct {char lexeme[30];tokenizer Token;int line_number;}tokenInfo;

This information required by the parser from the lexer is encapsulated in this structure.

[I]. Interface with parser: getNextToken() acts as an interface between the lexer and the parser.

7. **Parser Details:**

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

Grammar : Represented as an array of structures where each structure contains the head of a doubly linked list which stores the production rule.

Parse Table : Represented as a two dimensional array where the production rule to be used is identified by a pair of numbers. The first indicates the non terminal corresponding to the production and the second identifies the correct branch of the production rule.

Parse Tree and Stack Node : The stack node structure encapsulates necessary information about tree nodes required for building the parse tree.

[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 tree traversal has been done as mentioned in the mail.

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

i. Data structure for original grammar rules: An array of doubly linked lists.
ii. FIRST and FOLLOW sets computation automated: Yes
iii. Data structure for representing sets: Array of unsigned long long int (Bitmask)
iv. Time complexity of computing FIRST sets: O(|Non-Terminals|)
v. Name the functions (if automated) for computation of First and Follow sets:
ComputeFirstAndFollowSets(), unsigned long long int follow_set(), void first(int nt_enum)
vi. If computed First and Follow sets manually and represented in file/function (name that): N.A

[D]. Error Handling

i. Attempted (yes/ no): yes
ii. Printing errors (All errors/ one at a time) : All errors for the given test case are reported.
iii. Describe the types of errors handled: Lexical errors described by DFA and Syntactic errors caused due to sync and error entries in the parse table.
iv. Synchronizing tokens for error recovery (describe) :were constructed using follow sets.
v. Total number of errors detected in the given test case t6(with_syntax_errors).txt: 12 errors are reported.(2 lexical and 10 syntax)

8. **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

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

10. **Execution**

[A]. status (describe in maximum 2 lines): Properly working on all given test cases.

[B]. Execution time taken for

- t1.txt (in ticks) 537 and (in seconds): 0.000537
- t2.txt (in ticks) 395 and (in seconds): 0.000395
- t3.txt (in ticks) 366 and (in seconds): 0.000366
- t4.txt (in ticks) 536 and (in seconds): 0.000536
- t5.txt (in ticks) 649 and (in seconds): 0.000649
- t6.txt (in ticks) 976 and (in seconds): 0.000976

[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: No

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

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

13. Declaration: We, Anirudh Buvanesh, Anubhav Bansal, Aayush Agarwal, Manan Agarwal, Ashish Kumar 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 2016B4A70614P	Name: Anirudh Buvanesh
ID 2016B4A70824P	Name: Anubhav Bansal
ID 2016B5A70716P	Name: Aayush Agarwal
ID 2016B5A70607P	Name Manan Agarwal
ID 2016B4A70636P	Name: Ashish Kumar

Date: 24th February 2020