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

Group No.
9

Compiler Construction (CS F363)
II Semester 2022-23
Compiler Project (Stage-1 Submission)
Coding Details
(March 2, 2023)

---

1. IDs and Names of team members

Name: **ARCHAJ JAIN** ID: **2020A7PS0072P**

Name: **SARTHAK SHAH** ID: **2020A7PS0092P**

Name: **SIDDHARTH KHANDELWAL** ID: **2020A7PS0098P**

Name: **RISHI RAKESH SRIVASTAVA** ID: **2020A7PS0108P**

Name: **BHANUPRATAP RATHORE** ID: **2020A7PS1675P**

2. Mention the names of the Submitted files :

1 driver.c	7 hashTableDef.h	13 parser.c	18 setADT.h
2 doublyLinkedList.c	8 lexer.c	14 parser.h	19 setADT.c
3 doublyLinkedList.h	9 lexer.h	15 parserDef.h	20 stackADT.h
4 grammar.txt	10 lexerDef.h	16 parseTree.c	21 stackADT.c
5 hashTable.c	11 makefile	17 parseTree.h	22 removeComments.h
6 hashTable.h	12 remove_comments.c	23 coding_details(this file)	

3. Total number of submitted files: **23 (including this one)** (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** [Note: Files without names will not be evaluated]

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

6. Lexer Details:

[A]. Technique used for pattern matching: **FOLLOWED DFA TRANSITIONS AND RETRACTIONS**

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

SWITCH CASE

[C]. Keyword Handling Technique: **HASHING**

[D]. Hash function description, if used for keyword handling: **lenstring%2+ summation of ascii(char)*(1<<(ascii(char)%8)) with linear probing to handle collisions .**

[E]. Have you used twin buffers? (yes/ no) **YES**

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

[G]. Describe the lexical errors handled by you :

1) **LEXEME SIZE MORE THAN 20**
2) **NOT A VALID SYMBOL**

3) **NOT A VALID OPERATOR**

4) **NOT A VALID REAL NUM** .

[H].Data Structure Description for tokenInfo (in maximum two lines): struct consisting of Token Type which is an enum, Line no., lexeme which is a union of int, float, char string

[I]. Interface with parser : Parser calls get next token() function to access tokens one by one and generates corresponding parse tree.

7. Parser Details:

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

i. grammar : struct of symbol (LHS) and RHS which is a doublylinked list of symbols where symbol is union of enums(terminal and nonterminal) and consists of a bool flag.

ii. parse table : 2-D array of pointers pointing to RHS of grammar rules

iii. parse tree: (Describe the node structure also) Node consisting of symbol, parent ptr, child ptr, sibling ptr, prevsibling ptr, and token. Parse tree is ptr to root node.

iv. Parsing Stack node structure : Consisting of stacknode ptr next and corresponding TreeNode ptr

v. Any other (specify and describe) : Pair for reserved words and corresponding enum values.

[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

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

i. Data structure for original grammar rules : **Doubly linked list**

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

iii. Data structure for representing sets : **Boolean Array**

iv. Time complexity of computing FIRST sets : **O(num_nonterminals*num_rules)**

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

1) `void populateFirstSet(nonterminal nt1);`

2) `void populateFollowSet(nonterminal nt1);`

vi. If computed First and Follow sets manually and represented in file/function (name that) :
Computation of First and Follow Sets is automated

[D]. Error Handling

i. Attempted (yes/ no): **YES**

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

iii. Describe the types of errors handled

1) Lexical Errors mentioned above
2) Syntactic Errors :
    a) Terminal mismatch
    b) Rule not found for non-terminal and current token
    c) Tokens remaining but stack is empty
    d) Stack is non-empty but tokens finished.

iv. Synchronizing tokens for error recovery (describe) :

1) When top of stack is non-terminal which does not matches with the current token then **we used follow set of non terminal , semicolon and end as a synchronizing set.**

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

17 including all errors specified in t6.txt

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) :

Working correctly for all testcases provided.

[B]. Execution time taken for

- t1.txt (in ticks) : **2877** and (in seconds) : **0.002877**
- t2.txt (in ticks) : **2158** and (in seconds) : **0.002158**
- t3.txt (in ticks) : **1792** and (in seconds) : **0.001792**
- t4.txt (in ticks) : **2870** and (in seconds) : **0.002870**
- t5.txt (in ticks) : **3176** and (in seconds) : **0.003176**
- t6.txt (in ticks) : **3204** and (in seconds) : **0.003204**

[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):

1) Syntactic errors not separated by delimter like ";" and "end" is difficult to handle.
2) Due to above difficulty, parseTree skips some tokens related to syntactically incorrect ones.

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

13. Declaration: We, **ARCHAJ JAIN** , **SARTHAK SHAH** , **SIDDHARTH KHANDELWAL** , **RISHI RAKESH SHRIVASTAVA** & **BHANUPRATAP RATHORE** , declare that we have put our genuine efforts into 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 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 name below]

ID **2020A7PS0072P** Name: **ARCHAJ JAIN**
ID **2020A7PS0092P** Name: **SARTHAK SHAH**
ID **2020A7PS0098P** Name: **SIDDHARTH KHANDELWAL**
ID **2020A7PS0108P** Name: **RISHI RAKESH SHRIVASTAVA**
ID **2020A7PS1675P** Name: **BHANUPRATAP RATHORE**

Date: **01/03/2023**

Should not exceed 4 pages.