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

Compiler Construction (CS F363)
II Semester 2023-24
Compiler Project
Coding Details
(March 5, 2024)

Group Number
7

1. Team Members Names and IDs

ID: 2020B1A71605P	Name: Samarth Trivedi
ID: 2020B1A71612P	Name: Harshil Sethi
ID: 2020B1A70644P	Name: Nitish Sahani
ID: 2020B3A70786P	Name: Neel Dedhia
ID: 2020B3A70794P	Name: Harsh Rathi
ID: 2020B5A70604P	Name: Divyan Poddar

2. Mention the names of the Submitted files :

1 lexer.c	7 lexer.h	13 lexerDef.h
2 driver.c	8 parser.c	14 parser.h
3 parserDef.h	9 colorCode.h	15 stack_ADT.c
4 stack_ADT.h	10 tree_ADT.c	16 tree_ADT.h
5 follow.txt	11 terminals.txt	17 non_terminals.txt
6 matrix.csv	12 grammar.txt	18 first.txt
19 t1-t6.txt	20 testcase1-6.txt	

3. Total number of submitted files (including copy the pdf file of this coding details pro forma) : 34 (All files should be in ONE folder named as Group_#)

4. Have you compressed the folder as specified in the submission guidelines? (yes/no) **YES**

5. **Lexer Details:**
   - [A]. Technique used for pattern matching: **The DFA is implemented using a switch case and compared with the contents loaded in the Twin Buffer.**
   - [B]. Keyword Handling Technique: **Lookup in a hash table**
   - [C]. Hash function description, if used for keyword handling: **We are doing chaining to avoid collisions using hashValue = 31 * hashValue + key[i]**
   - [D]. Have you used twin buffer? (yes/ no): **YES**
   - [E]. Error handling and reporting (yes/No): **YES**
   - [F]. Describe the errors handled by you: **Errors related to Unknown patterns, Errors related to exceeding max length of identifiers**
   - [G]. Data Structure Description for tokenInfo (in maximum two lines): **It is a struct that contains token_type(which is an enum of all the token types in the language), lexeme, line number and a boolean isEoF.**

6. **Parser Details:**

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

i. grammar : Chaining using linked list - **Every non-terminal points to a rule. A rule points to next rule, and the terminals and non-terminals in the RHS of the grammar. All grammar is stored in an array of pointers of type rule***

ii. FIRST and FOLLOW sets - **Array of unsigned long long int - bits in the unsigned long long num represents 0/1 for each Terminal. We have used bit masking and bit manipulation to compute union of sets.**

iii. parse table: **Matrix of type rule* of size NON-TERMINALS x TERMINALS.**

iv. parse tree: (Describe the node structure also): **The TreeNode struct represents a node in a tree data structure. It contains a pointer to the terminals/non-terminals, along with pointers to its first child, next sibling, and parent nodes. Additionally, it stores a line number and lexical information.**

v. Any other (specify and describe) Custom follow ADT for intermediate storage during follow computation, and also for representation.

[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 - Leftmost child --> parent node--> remaining siblings (excluding the leftmost)**

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

i. Data structure for original grammar rules: **Linked lists chaining for all rules - struct used rule* and variable***

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

iii. Name the functions (if automated) for computation of First and Follow sets **populateFirst(); populateFollow();**

iv. If computed First and Follow sets manually and represented in file/function (name that) **Automated**

[D]. Error Handling

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

vi. Describe the types of errors handled:

Following cases are handled:
i. When stack top is terminal and does not match with incoming token.
ii. When Stack top is non-terminal and incoming token corresponds to TK_ERROR. Skip incoming token
iii. SYN case is handled. Additional care is taken to enter the tokens which are in the beginning of new line to take it as a SYN case. So, the errors are not carry forwarded to next line. Recovery is ASAP.

7. 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**

8. Driver Details: Does it take care of the options specified earlier(yes/no): **YES**
9. Execution
   [A]. status (describe in maximum 2 lines): **The code is running perfectly fine on all the given test cases as well as on multiple other test cases.**
   [B]. Gives segmentation fault with any of the test cases (1-6) uploaded on the course page. If yes, specify the testcase file name: **NO**

10. Specify the language features your lexer or parser is not able to handle (in maximum one line): **NONE**

11. Are you availing the lifeline (Yes/No): **YES**

12. Declaration: We, Samarth Trivedi, Harshil Sethi, Nitish Sahani, Neel Dedhia, Harsh Rathi, Divyan Poddar declare that we have put our genuine efforts in creating the compiler project code and have submitted the code developed only by us. 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 all of us in our team and we will accept the penalty as decided by the department of Computer Science and Information Systems, BITS, Pilani.

ID: 2020B1A71605P	Name: Samarth Trivedi
ID: 2020B1A71612P	Name: Harshil Sethi
ID: 2020B1A70644P	Name: Nitish Sahani
ID: 2020B3A70786P	Name: Neel Dedhia
ID: 2020B3A70794P	Name: Harsh Rathi
ID: 2020B5A70604P	Name: Divyan Poddar

Date: 6/3/2024

Not to exceed 3 pages.