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, 2022)

Group Number
8

---

1. Team Members Names and IDs

ID	2020B1A70630P	Name	Aditya Thakur
ID	2021A7PS2001P	Name	Amal Sayeed
ID	2021A7PS2005P	Name	Ohiduz Zaman
ID	2021A7PS2682P	Name	Priyansh Patel
ID	2021A7PS2002P	Name	Rachoita Das
ID	2020B1A70611P	Name	Subhramit Basu Bhowmick

2. Mention the names of the Submitted files :

1 grammar.txt	7 parser.h	13 testcase2.txt
2 coding_details.pdf	8 parser.c	14 testcase3.txt
3 lexerDef.h	9 driver.c	15 testcase4.txt
4 lexer.c	10 makefile	16 testcase5.txt
5 lexer.h	11 stack.h	17 testcase6.txt
6 parserDef.h	12 testcase1.txt	

3. Total number of submitted files (including copy the pdf file of this coding details pro forma) : 17 (All files should be in ONE folder named as Group_8)

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

5. **Lexer Details:**
   [A]. Technique used for pattern matching: Used a switch-case statement in a loop to implement the DFA with transitions as shown in the DFA submission. (In getNextToken function of lexer.c)
   [B]. Keyword Handling Technique: Implemented a trie data structure (fast lookup of words) for populating it with keywords, and whenever a fieldid pattern is matched, lookup in the trie, and if present, then tokenize as keyword, otherwise tokenize as fieldid.
   [C]. Hash function description, if used for keyword handling: Did not use hash function. Instead used a trie (as mentioned above) which tells about presence/absence of a word in O(length_of_word) time.
   [D]. Have you used twin buffer? (yes/ no) : Yes
   [E]. Error handling and reporting (yes/No): Yes
   [F]. Describe the errors handled by you: Handled all cases including unrecognized pattern, variable name or function name exceeding max length.
   [G]. Data Structure Description for tokenInfo (in maximum two lines): A struct containing line number and a pointer to symbol table entry

6. **Parser Details:**

[A]. High Level Data Structure Description (in maximum three lines each, avoid giving C definitions used):
   i. grammar : An array of structs; Each struct/production rule has an LHS of type "grammar symbol" and an RHS of type "linked list of grammar symbols."
   ii. FIRST and FOLLOW sets: An array of linked lists where each list corresponds to a first set or a follow set

iii. parse table: A 2-D array or matrix with non terminals as first dimension index and terminals as second dimension index and stored element is a pointer to a grammar rule.
iv. parse tree: (Describe the node structure also): An n-ary tree with a specified root. Each node of the tree has a grammar symbol (terminal / non-terminal), pointer to symbol table entry (for terminals), an array of pointers to children nodes, and a count of number of children

[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): Printed as per the inorder traversal of the tree i.e., for a given node, print the subtree rooted at first child, then print the current node, then the other children.

[C]. Grammar and Computation of First and Follow Sets
  i. Data structure for original grammar rules: An array of structs with LHS as grammar symbol and RHS as a linked list of grammar symbols
  ii. FIRST and FOLLOW sets computation automated (yes /no): Yes
  iii. Name the functions (if automated) for computation of First and Follow sets: computeFirstSets() and computeFollowSets()
  iv. If computed First and Follow sets manually and represented in file/function (name that) : Did NOT do manually

[D]. Error Handling
  v. Attempted (yes/ no): Yes
  vi. Describe the types of errors handled: All syntactic errors that do not conform to the given grammar are reported, such as, unexpected token or missing tokens.

7. Compilation Details:
  [A]. Makefile works (yes/no): Yes
  [B]. Code Compiles (yes/ no): Yes
  [C]. Mention the .c files that do not compile: All files compile successfully
  [D]. Any specific function that does not compile: All functions compile successfully
  [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): Fully functional without any errors/faults.
  [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 segmentation faults

10. Specify the language features your lexer or parser is not able to handle (in maximum one line): All features work as expected

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

12. Declaration: We, Aditya Thakur, Amal Sayeed, Ohiduz Zaman, Priyansh Patel, Rachoita Das, Subhramit Basu Bhowmick 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.

Your names and IDs

Name: Aditya Thakur	ID: 2020B1A70630P
Name: Amal Sayeed	ID: 2021A7PS2001P
Name: Ohiduz Zaman	ID: 2021A7PS2005P
Name: Priyansh Patel	ID: 2021A7PS2682P
Name: Rachoita Das	ID: 2021A7PS2002P
Name: Subhramit Basu Bhowmick	ID: 2020B1A70611P

Date: 05/03/2024 (DD/MM/YYYY)

Not to exceed 3 pages.