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
11

1. Team Members Names and IDs
   - ID 2021A7PS0450P Name Rudra Jewalikar
   - ID 2021A7PS0008P Name Dhruv Shrimali
   - ID 2021A7PS2004P Name Salil Godbole
   - ID 2021A7PS0013P Name Shyam Raghavan
   - ID 2021A7PS2535P Name Sarthak Sharma

2. Mention the names of the Submitted files:
   1. TD.c
   2. TD.h
   3. parserDef.c
   4. parserDef.h
   5. grammarParser.h
   6. grammarParser.c
   7. lexer.h
   8. lexer.c
   9. parseTable.c
   10. parseTable.h
   11. parser.c
   12. parser.h
   13. parseTree.h
   14. parseTree.c
   15. stack.c
   16. stack.h
   17. lexerDef.c
   18. lexerDef.h
   19. driver.c
   20. makefile
   21. Coding Details
   22. Grammar.txt
   23. testcase1.txt
   24. testcase2.txt
   25. testcase3.txt
   26. testcase4.txt
   27. testcase5.txt
   28. testcase6.txt
   29. testcase7.txt

3. Total number of submitted files (including copy of the pdf file of this coding details pro format): 29
   (All files should be in ONE folder named 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: DFA

[B]. Keyword Handling Technique: Symbol Table

[C]. Hash function description, if used for keyword handling: Key is the string itself, returns zero if not present, and it gets initialized before lexer starts.

[D]. Have you used twin buffer? (yes/ no): Yes

[E]. Error handling and reporting (yes/No): Yes

[F]. Describe the errors handled by you: The length of the token is longer than 20 characters, the token does not match the pattern, unidentified symbol

[G]. Data Structure Description for tokenInfo (in maximum two lines): token name, lexeme, line number occurrence, integer value if int, float value if float.

6. Parser Details:

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

i. grammar: Structure for Grammar stores all the terminal, non-terminal symbols and production rules along with their number. The structure describing the production rules stores a single symbol in LHS of the rule and symbols in RHS separately. The structure for the symbol only stores its name.

ii. FIRST and FOLLOW sets: We have designed a Hash Table which stores a set of symbols(value) corresponding with each Non-Terminal(key).

iii. parse table: The same hash table (which was used for first and follow) is used but the key is a combination of both Non-terminal and terminal. Only symbols in the RHS of the corresponding rule are stored(in a way storing the entire rule). If there is no value corresponding to a key means there is no table entry(no rule).

iv. parse tree: (Describe the node structure also) : Each node contains an array of pointers to child nodes and number of child nodes. If a node is leaf, it also contains values corresponding to the lexeme contained in that node such as the lexeme, line number and int or float value. These values are stored in a separate node which is only assigned space for leaf nodes thus saving space.

[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

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

i. Data structure for original grammar rules:

```
typedef struct {
 Symbol left;                // Left-hand side non-terminal
 Symbol right[MAX_RULELENGTH]; // Right-hand side symbols
 int numSymbols;             // Number of symbols on the right hand side
ProductionRule;
```

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

iii. Name the functions (if automated) for computation of First and Follow sets: ComputeFirstAndFollowSets

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

[D]. Error Handling

i. Attempted (yes/ no): Yes

ii. Describe the types of errors handled: Terminal mismatches were handled by skipping tokens on the token stream till a ‘ ; ’ appears. Non-terminal rule mismatches were handled by skipping the input till we find a relevant token that is present in the sync set and then popping the Non-Terminal from stack.

7. 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: No
   [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):
   Functionalities 0-4 all work, with test cases 1-6. Extra testcase “testcase7.txt” also provided.

   [B]. Gives segmentation fault with any of the test cases (1-6) uploaded on the course page. If yes, specify the test case file name: NA

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

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

12. Declaration: We, Shyam Raghavan, Rudra Jewailker, Dhruv Shrimali, Salil Godbole, Sarthak Sharma 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

ID 2021A7PS0450P Name Rudra Jewalikar
ID 2021A7PS0008P Name Dhruv Shrimali
ID 2021A7PS2004P Name Salil Godbole
ID 2021A7PS0013P Name Shyam Raghavan
ID 2021A7PS2535P Name Sarthak Sharma

Date: 5/3/2024
