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

Compiler Construction (CS F363)
II Semester 2021-22
Compiler Project (Stage-1 Submission)
Coding Details
(March 3, 2022)

Group Number
13

1. Team Members Names and IDs
   ID 2018B5A70423P Name Ashwin Kiran Godbole
   ID 2018B2A70362P Name Samarth Krishna Murthy

2. Mention the names of the Submitted files :

1 grammar.txt	7 parser.h	13 testcase3.txt	19 testcase9.txt
2 coding details stage1.pdf	8 parser.c	14 testcase4.txt	
3 lexerDef.h	9 driver.c	15 testcase5.txt	
4 lexer.c	10 makefile	16 testcase6.txt	
5 lexer.h	11 testcase1.txt	17 testcase7.txt	
6 parserDef.h	12 testcase2.txt	18 testcase8.txt	

3. Total number of submitted files (including copy the pdf file of this coding details pro forma) : 19 (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: Transition Diagram
   [B]. Keyword Handling Technique: Hashtable and searchToken() function
   [C]. Hash function description, if used for keyword handling: djb2 hashfunction was used. (http://www.cse.yorku.ca/~oz/hash.html)
   [D]. Have you used twin buffer? (yes/ no) **no**
   [E]. Error handling and reporting (yes/No) **yes**
   [F]. Describe the errors handled by you : Lexical errors and Syntactic errors (with error recovery).
   [G]. Data Structure Description for tokenInfo (in maximum two lines): every 'token' has:
   (1) 64-bit integer to store line (2) enumeration value to indicate type of the token
   (3) a union that can store either an integer, a double or a char pointer

6. **Parser Details:**

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

i. grammar: Array of structures of type production rules (prodRule) and an integer indicating number of rules in the given grammar.

ii. FIRST and FOLLOW sets: Structure containing enum value for non-terminal, arrays of token types for FIRST set and FOLLOW set, the number of items in FIRST set, the number of items in FOLLOW set, a field to indicate if the FIRST set has 'epsilon', a field to indicate if the FOLLOW set contains '$' and a field to indicate if the FIRST set has been calculated.

iii. parse table: Two 2-D arrays, one for storing production rules and the other to indicate existence of production rule at corresponding index.

iv. parse tree: (Describe the node structure also) : A node of the parse tree consists of two integers to indicate whether it is a terminal/non-terminal and to store the depth of the node. Pointers are present which point to

the nodes' parent, its left child and its right siblings. The tree is built as and when the stack is popped and the nodes are added to the tree based on the depth.

v. prodRule : A structure containing a rule in the form of a non-terminal and its expansion. This structure was built to ease the production of first and follow sets.
vi. parseStack: A structure containing the list of non-terminals/terminals in the stack that are to be expanded/popped.

[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 of nodes (left->node->right)

[C]. Grammar and Computation of First and Follow Sets
   i. Data structure for original grammar rules: data structure name: **gram**
   ii. FIRST and FOLLOW sets computation automated (yes /no): **yes**
   iii. Name the functions (if automated) for computation of First and Follow sets: computeFirsts, computeFollows, first, follow
   iv. If computed First and Follow sets manually and represented in file/function (name that): N/A

[D]. Error Handling
   v. Attempted (yes/ no): **yes**
   vi. Describe the types of errors handled: **Lexical errors**: Error reported and illegal patterns/characters skipped. **Syntax errors**: If terminal on the top of the stack and the current terminal in the token stream do not match, stack top is popped. Else if no rule exists in parse table for non-terminal stack top and token in token stream then tokens are skipped until a token belonging to the synch set of stack top is found. (**Panic mode**)

7. 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**

8. Driver Details: Does it take care of the options specified earlier(yes/no): **yes**
9. Execution
   [A]. status (describe in maximum 2 lines): Lexical and syntax analysis working correctly. Attempted to generate parse tree file according to format specified
   [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 fault**

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

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

12. Declaration: We, Ashwin Kiran Godbole and Samarth Krishna Murthy (your names) 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 Ashwin Kiran Godbole ID 2018B5A70423P
Name Samarth Krishna Murthy ID 2018B2A70362P

Date: 3 March, 2022

Not to exceed 3 pages.