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 4, 2022)

Group Number

15

1. Team Members Names and IDs

ID 2019A7PS0077P	Name Aadit Deshpande
ID 2019A7PS0097P	Name Nandan B Parikh
ID 2019A7PS0088P	Name Preetika Verma
ID 2019A7PS1140P	Name Pritika Ramu
ID 2019A7PS0042P	Name Sneha

2. Mention the names of the Submitted files :

1 driver.c	8 parser.h	15 First.txt
2 lexer.c	9 parserDef.h	16 Follow.txt
3 lexerDef.h	10 makefile	17 ReadableFirst.txt
4 lexer.h	11 stack.c	18 ReadableFollow.txt
5 symbolTable.c	12 predParseTable.c	19-24 testcase1.txt…testcase6.txt
6 grammar.txt	13 ll1Grammar.c	
7 parser.c	14 coding details stage 1.pdf	

3. Total number of submitted files (including copy the pdf file of this coding details pro forma) : **24**
   (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: **Using Regular expressions (Implemented by a DFA) and backtracking**
   [B]. Keyword Handling Technique: **Initializing the keywords in Symbol Table (uses Chained Hashing technique)**
   [C]. Hash function description, if used for keyword handling:
   h(s)=(7+31*s[0]+(31^2)*s[1]+…+(31^n+1)*s[n])%SIZE
   Where, s is a string of length n, and SIZE is size of the hash table
   [D]. Have you used twin buffer? (yes/ no) **YES**
   [E]. Error handling and reporting (yes/No): **YES**
   [F]. Describe the errors handled by you: **Lexical Errors (Identifier length exceeded, illegal characters, spelling errors for different tokens)**
   [G]. Data Structure Description for tokenInfo (in maximum two lines): **contains Enum value for the Terminal/Non Terminal, corresponding Lexeme, Numerical values (numbers only), the token's Line No.**

6. **Parser Details:**

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

i. Grammar: **Array of Linked Lists, each pointing to Linked Lists. Each Array element represents a Non-Terminal. Each element also has the head of a Linked list with all its Alternate Productions. Each Production is stored as a Linked List of terminals and non-terminals (differentiated using a Boolean).**

ii. FIRST and FOLLOW sets: **2D Bit vectors (Rows represent Non-terminals and columns are terminals)**

iii. parse table: **2D Array of Linked Lists (Each linked list represents a Grammar Rule)**

iv. parse tree: (Describe the node structure also): **The Parse tree is a pointer to the struct 'tree*'. Each 'tree' (node) struct contains the enum value for its symbol (Terminal/Non-terminal), the line no. for the token and pointers to its Parent, First Child and immediate right sibling**

v. Any other (specify and describe): **The stack is implemented using a linked list of 'rule' structures (The same data structures used in the Grammar)**

[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) <u>**In-Order**</u>**: The parse tree is an N-ary tree, so we print the children in left-to-right order (1st to N-1th), then print the node and then print the rightmost child**

[C]. Grammar and Computation of First and Follow Sets
- i. Data structure for original grammar rules: **Array of Linked Lists, each pointing to Linked Lists**
- ii. FIRST and FOLLOW sets computation automated (yes /no): **YES (Partially)**
- iii. Name the functions (if automated) for computation of First and Follow sets: **computeFirstandFollow() automates First Set computation, but not yet follow set**
- iv. If computed First and Follow sets manually and represented in file/function (name that): **ReadableFirst.txt, ReadableFollow.txt**

[D]. Error Handling
- v. Attempted (yes/ no): **YES**
- vi. Describe the types of errors handled: **Syntactical errors (like Unmatched parenthesis, missing semicolons, incorrect declarations and so on.**

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): **Compiles and runs without any error or warnings**
   - [B]. Gives segmentation fault with any of the test cases (1-6) uploaded on the course page. If yes, specify the testcase file name: **None**

10. Specify the language features your lexer or parser is not able to handle (in maximum one line)
    Automation of computation of Follow Set, Could implemented error recovery but with partial Synch set

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

12. Declaration: We, Pritika Ramu, Preetika Verma, Sneha, Nandan B Parikh and Aadit Deshpande 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: Sneha	ID: 2019A7PS0042P
Name: Pritika Ramu	ID: 2019A7PS1140P
Name: Nandan B Parikh	ID: 2019A7PS0097P
Name: Preetika Verma	ID: 2019A7PS0088P
Name: Aadit Deshpande	ID: 2019A7PS0077P

Date: 4 March 2022

Not to exceed 3 pages.