**BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI**

**DEPARTMENT OF COMPUTER SCIENCE AND INFORMATION SYSTEMS**

**Compiler Construction (CS F363)**

**Group No.**

**19**

**II Semester 2022-23**

**Compiler Project (Stage-1 Submission)**

**Coding Details**

**(March 2, 2023)**

1. IDs and Names of team members

ID: 2019B3A70545P Name: ANIMESH BHARGAVA

ID: 2019B5A70226P Name: ISHVIT BHASIN

ID: 2019B2A70878P Name: RITIK THAKUR

ID: 2019B3A70154P Name: SAHIL GUPTA

ID: 2019B4A70704P Name: UTKARSH YASHVARDHAN

1. Mention the names of the Submitted files :

1\_\_\_\_\_\_**driver.c**\_\_\_\_\_\_\_ 7\_\_\_\_\_**grammar.txt**\_\_\_ 13\_\_\_\_\_**stack.c**\_\_\_\_\_

2\_\_\_\_\_\_**lexer.c**\_\_\_\_\_\_\_\_ 8\_\_\_\_\_**hashDef.h**\_\_\_\_\_\_ 14\_\_\_\_\_**stack.h**\_\_\_\_\_

3\_\_\_\_\_\_**lexerDef.h**\_\_\_\_\_ 9\_\_\_\_\_**hashtable.c**\_\_\_\_\_ 15\_\_\_\_**makefile \_**\_\_\_\_

4\_\_\_\_\_\_**parser.c**\_\_\_\_\_\_\_ 10\_\_\_\_**hashtable.h**\_\_\_\_ 16\_\_\_\_\_ **tree.h**\_\_\_\_\_\_

5\_\_\_\_\_\_**parser.h** \_\_\_\_\_\_ 11\_\_\_\_**set.c**\_\_\_\_\_\_\_\_\_\_ 17\_\_\_\_\_**tree.c\_**\_\_\_\_

6\_\_\_\_\_\_**parseDef.h**\_\_\_\_ 12\_\_\_\_**set.h\_\_**\_\_\_\_\_\_\_\_

TEST CASE FILES :  **1. t1.txt 2. t2.txt 3. t6.txt**

1. Total number of submitted files: **20**  (All files should be in **ONE folder** named exactly as Group\_#, # is your group number)
2. Have you mentioned your names and IDs at the top of each file (and commented well)? **YES**  [ Note: Files without names will not be evaluated]
3. Have you compressed the folder as specified in the submission guidelines? **YES**
4. **Lexer Details:**
   1. Technique used for pattern matching: **DFA built using switch cases and supplementary functions.**
   2. DFA implementation (State transition using switch case, graph, transition table, any other (specify): **State transition using switch case, supplementary functions, twin buffer.**
   3. Keyword Handling Technique: **Hash table and twin buffer.**
   4. Hash function description, if used for keyword handling: **Polynomial hash function with double hashing**
   5. Have you used twin buffer? **YES**
   6. Lexical error handling and reporting: **YES**
   7. Describe the lexical errors handled by you : **LEX\_ERROR\_1 which reports if a lexeme (which is an ID) length exceeds 20, and LEX\_ERROR\_2 which reports an invalid lexeme.**
   8. Data Structure Description for tokenInfo (in maximum two lines): **TOKEN data structure has been created. It has the fields: a char array “lexeme” to store the lexeme token id, then an integer to store the line number of the token, and finally a union which stores the value of the lexeme which should be an ID, integer number or float number.**
   9. Interface with parser : **The function “getNextToken()” from lexer sends tokens to the function “parseUserSourceCode()” of parser.**
5. **Parser Details:** 
   1. **High Level Data Structure Description (in maximum three lines each, avoid giving C definitions used):**
      1. grammar : **A 1D array of linked lists. The first node represents the LHS of the grammar rule and the subsequent nodes represent the RHS nodes from left to right.**
      2. parse table : **A 2D array where each row corresponds to a non terminal and the row represents the terminals. If a terminal is in the parse table row of a non-terminal, it is represented by the value 1.**
      3. parse tree: (Describe the node structure also) **The node of our parse tree is called “grammarSymbol”. It has a boolean to denote whether it is a terminal or non terminal, as well as a union for storing the terminal or non terminal enum value.  
         The parse tree is built using pointers and linked lists. A root node points to the left most child, which holds the pointer to the right nodes of the tree. From each node, further children can be pointed to.**
      4. Parsing Stack node structure : **A linked list of stack nodes. The initial stack node pointer points to the first node. Each stack node has a symbol field, a pointer to the previous symbol and a pointer to the next symbol. If no next symbol exists, NULL is pointed to.**
      5. Any other (specify and describe) : **NA**
   2. **Parse tree** 
      1. Constructed : **YES**
      2. Printing as per the given format : **YES**
      3. Describe the order you have adopted for printing the parse tree nodes (in maximum two lines) : **Inorder printing. Left most child followed by parent followed by remaining right children.**
   3. **Grammar and Computation of First and Follow Sets** 
      1. Data structure for original grammar rules : **We have created a “rules” 1D array of linked lists. Each linked list represents a rule, the first node is the LHS, and the remaining node are the RHS terms from left to right.**
      2. FIRST and FOLLOW sets computation automated : **YES.**
      3. Data structure for representing sets : **A 1D boolean array.**
      4. Time complexity of computing FIRST sets **O(TOTAL\_RULES x TOTAL\_NON\_TERMINALS x TOTAL\_TERMINALS)**
      5. Name the functions (if automated) for computation of First and Follow sets : **firstSetOfRule(), generateFirstSets(), generateFollowSet().**
      6. If computed First and Follow sets manually and represented in file/function (name that) : **N/A.**
   4. **Error Handling** 
      1. Attempted : **YES**
      2. Printing errors (All errors/ one at a time) : **One at a time**
      3. Describe the types of errors handled : **Stack empty but input lexer stream not empty, stack top and input lexeme don’t match, no corresponding parse table rule for an input lexeme, full user code parsed by stack is not empty.**
      4. Synchronizing tokens for error recovery (describe) **SEMICOL, and END and FollowSet of stack top.**
      5. Total number of errors detected in the given testcase t6(with\_syntax\_errors).txt: **11**
6. **Compilation Details:**
   1. Makefile works : **YES**
   2. Code Compiles : **YES**
   3. Mention the .c files that do not compile:\_\_\_**NA**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
   4. Any specific function that does not compile:\_\_\_**NA**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
   5. Ensured the compatibility of your code with the specified gcc version: **YES**
7. **Driver Details**: Does it take care of the options specified earlier : **YES**
8. **Execution** 
   1. status (describe in maximum 2 lines): **All the function are executing with correct output for the provided test case files.**
   2. Execution time taken for
      * t1.txt (in ticks) \_\_**2524.00**\_\_\_\_\_\_ and (in seconds) \_\_\_\_**0.002524 sec**\_\_\_\_\_\_\_\_
      * t2.txt (in ticks) \_\_**2744.00**\_\_\_\_\_\_ and (in seconds) \_\_**0.002744 sec**\_\_\_\_
      * t3.txt (in ticks) \_\_**3005.00** \_\_\_\_\_ and (in seconds) \_\_\_**0.003005 sec**\_\_\_\_\_\_\_\_
      * t4.txt (in ticks) \_\_**6158.00**\_\_\_\_\_\_ and (in seconds) \_\_**0.006158 sec**\_\_\_\_\_\_\_\_
      * t5.txt (in ticks) \_\_**4622.00**\_\_\_\_\_\_ and (in seconds) \_\_**0.004622 sec**\_\_\_\_\_\_
      * t6.txt (in ticks) \_\_\_**2810.00**\_\_\_\_\_ and (in seconds) \_\_**0.002810 sec**\_\_\_\_\_\_\_
   3. Gives segmentation fault with any of the test cases (1-6) uploaded on the course page. If yes, specify the testcase file name:\_\_**NA**\_\_\_\_
9. Specify the language features your lexer or parser is not able to handle (in maximum one line)\_\_**NA**\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
10. Are you availing the lifeline : **YES**
11. Declaration: We, ANIMESH BHARGAVA, ISHVIT BHASIN, RITIK THAKUR, SAHIL GUPTA, and UTKARSH YASHVARDHAN, declare that we have put our genuine efforts into creating the compiler project code and have submitted the code developed only by our group. 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 disciplinary action as per the institute rules will be taken against us and we will accept the penalty as decided by the department of Computer Science and Information Systems, BITS, Pilani. [Write your ID and name below]

ID : 2019B3A70545P Name: ANIMESH BHARGAVA

ID : 2019B5A70226P Name: ISHVIT BHASIN

ID : 2019B2A70878P Name: RITIK THAKUR

ID : 2019B3A70154P Name: SAHIL GUPTA

ID : 2019B4A70704P Name: UTKARSH YASHVARDHAN

Date: 3rd March, 2023

---------------------------------------------------------------------------------------------------------------------------------------------

Should not exceed 4 pages.