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

**DEPARTMENT OF COMPUTER SCIENCE AND INFORMATION SYSTEMS**

**Compiler Construction (CS F363)**

**II Semester 2019-20**

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

**Coding Details** 

**(February 24, 2020)**

1. IDs and Names of team members

ID: 2017A7PS0136P Name: VITTHAL BHANDARI

ID: 2017A7PS0180P Name: IESHAAN SAXENA

ID: 2017A7PS0181P Name: NAMAN SINGHAL

ID: 2017A7PS0233P Name: DARSHAN AGRAWAL

ID: 2017A7PS0236P Name: ABHISHEK GUPTA

1. Mention the names of the Submitted files :

1. lexerDef.h 7. driver.c 13. t5.txt

2. lexer.h 8. makefile 14. t6.txt

3. lexer.c 9. t1.txt 15. grammar.txt

4. parserDef.h 10. t2.txt 16. proforma

5. parser.h 11. t3.txt 17.

6. parser.c 12. t4.txt 18.

1. Total number of submitted files: **16** (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/ no) **YES**

[Note: Files without names will not be evaluated]

1. Have you compressed the folder as specified in the submission guidelines? (yes/no) **YES**
2. **Lexer Details:**
   1. Technique used for pattern matching: We read the first character, then according to the DFA we read the further characters and reached to the conclusions accordingly.
   2. DFA implementation (State transition using switch case, graph, transition table, any other (specify): State transitions implemented using switch case
   3. Keyword Handling Technique: An array of keywords with each keyword hashed into a hash table; check each identifier for keyword by hashing it and comparing it with already stored keyword
   4. Hash function description, if used for keyword handling: The hash function adds the ASCII value of each character of the keyword token into a sum and computes hash value as -> sum mod 107
   5. Have you used twin buffer? (yes/ no) : **NO**
   6. Lexical error handling and reporting (yes/No): **YES**
   7. Describe the lexical errors handled by you : Incorrect identifier not following the rules has been classified as a lexical error( such as id length>20, NOT beginning with an alphabet, containing symbols other than alphabets, digits and underscore are all violations of the rules, hence errors)
   8. Data Structure Description for tokenInfo (in maximum two lines): The structure tokenInfo contains strings for lexeme and token, integer for line number, a union “value” which stores literal value of natural/real number for the respective case and a pointer to the next token(for further use)
   9. Interface with parser : the parser invokes the getToken() function to obtain next token from the lexer
3. **Parser Details:** 
   1. **High Level Data Structure Description (in maximum three lines each, avoid giving C definitions used):**
      1. grammar : Grammar is an array of rules where each rule is a linked list having first node as the LHS and remaining nodes as RHS of the rule in order.
      2. parse table : **YES**
      3. parse tree: (Describe the node structure also) : Tree is formulated as a node with many children. Leftmost children is the child and rest are siblings of that child . Each node in the stack contains strings for storing terminal and non-terminal and an integer tag for specifying what that node is (1 for terminal and 0 for Non-terminal). It contains a pointer to its child and another pointer to its immediate sibling.
      4. Parsing Stack node structure: This structure contains a pointer to a node(used in tree) for its use like actual stack and a pointer to another stack.
      5. Any other (specify and describe) : FIRST and FOLLOW sets have been implemented as an array where each array element non terminal string and a pointer to first set and follow set. Each set is implemented as a linked list.
   2. **Parse tree** 
      1. Constructed (yes/no): **YES**
      2. Printing as per the given format (yes/no): **YES**
      3. Describe the order you have adopted for printing the parse tree nodes (in maximum two lines):

Inorder traversal of n-ary parse tree as leftmost child->Parent->Rest Siblings

* 1. **Grammar and Computation of First and Follow Sets** 
     1. Data structure for original grammar rules: Each rule is a structure with a string and pointer to the next structure.
     2. FIRST and FOLLOW sets computation automated (yes /no): **YES**
     3. Data structure for representing sets: each element of the set contains a string for storing terminal and a pointer to the next element of that set. Each element also stores a line number for referring to the rule it derived from in corresponding grammar
     4. Time complexity of computing FIRST sets: Θ(n2)
     5. Name the functions (if automated) for computation of First and Follow sets:

First Sets -> computeFirst() , findFirst() , remove\_empty() , union\_func() , contains(), add()

Follow Sets -> computeFollow () , findFollow() , remove\_empty() , union\_func() , contains()

* + 1. If computed First and Follow sets manually and represented in file/function (name that): **N/A**
  1. **Error Handling** 
     1. Attempted (yes/ no): **YES**
     2. Printing errors (All errors/ one at a time) : **YES**
     3. Describe the types of errors handled: Lexical errors(incorrect lexemes due to violation of identifier naming convention) are identified by lexer while syntax errors by the parser.
     4. Synchronizing tokens for error recovery (describe) : (1) if top of the stack is non-terminal and the parse table corresponding to this non-terminal and the i/p symbol contains EMPTY , then discard the i/p symbols until a symbol comes in the follow set of the non-terminal. After this discard the non-terminal. (2) ) if top of the stack is terminal and it does not match with the i/p symbol, simply discard the top of the stack.
     5. Total number of errors detected in the given testcase t6(with\_syntax\_errors).txt : **18**

1. **Compilation Details:**
   1. Makefile works (yes/no): **YES**
   2. Code Compiles (yes/ no): **YES**
   3. Mention the .c files that do not compile : **N/A**
   4. Any specific function that does not compile : **N/A**
   5. Ensured the compatibility of your code with the specified gcc version(yes/no) : **YES**
2. **Driver Details**: Does it take care of the options specified earlier(yes/no) : **YES**
3. **Execution** 
   1. status (describe in maximum 2 lines):\_Perfect\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* 1. Execution time taken for
     + t1.txt (in ticks) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ and (in seconds) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
     + t2.txt (in ticks) \_\_\_10628\_\_\_\_\_\_\_\_\_\_\_\_\_ and (in seconds) \_\_0.01268\_\_\_\_\_\_\_\_\_\_\_\_
     + t3.txt (in ticks) \_\_\_29140\_\_\_\_\_\_\_\_\_\_\_\_\_ and (in seconds) \_\_0.02914\_\_\_\_\_\_\_\_\_\_\_\_
     + t4.txt (in ticks) \_\_\_65396\_\_\_\_\_\_\_\_\_\_\_\_\_ and (in seconds) \_\_0.065396\_\_\_\_\_\_\_\_\_\_\_
     + t5.txt (in ticks) \_\_\_61361\_\_\_\_\_\_\_\_\_\_\_\_\_ and (in seconds) \_\_\_0.061361\_\_\_\_\_\_\_\_\_\_
     + t6.txt (in ticks) \_\_\_93534\_\_\_\_\_\_\_\_\_\_\_\_\_ and (in seconds) \_\_0.093534\_\_\_\_\_\_\_\_\_\_\_
  2. Gives segmentation fault with any of the test cases (1-6) uploaded on the course page. If yes, specify the testcase file name :  **t1.txt, (doesn’t work with parser only, otherwise works fine with lexer)**

1. Specify the language features your lexer or parser is not able to handle (in maximum one line) : **N/A**
2. Are you availing the lifeline (Yes/No) **: YES**
3. Declaration: We, Vitthal Bhandari, Iehaan Saxena, Naman Singhal, Darshan Agrawal and Abhishek Gupta declare that we have put our genuine efforts in 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 a 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.

ID: 2017A7PS0136P Name: VITTHAL BHANDARI

ID: 2017A7PS0180P Name: IESHAAN SAXENA

ID: 2017A7PS0181P Name: NAMAN SINGHAL

ID: 2017A7PS0233P Name: DARSHAN AGRAWAL

ID: 2017A7PS0236P Name: ABHISHEK GUPTA

Date: 24/02/2020

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