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

**DEPARTMENT OF COMPUTER SCIENCE AND INFORMATION SYSTEMS**

**Compiler Construction (CS F363)**

**II Semester 2022-23**

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

**Coding Details**

**Group No.**

4

**(March 2, 2023)**

1. IDs and Names of team member

ID:\_\_\_\_\_2020A7PS0123P\_\_\_\_\_\_\_\_\_\_\_\_Name:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Aryan Desai\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ID:\_\_\_\_\_2020A7PS0979P\_\_\_\_\_\_\_\_\_\_\_\_Name:\_\_\_\_\_\_\_\_\_\_Ramakant Pandurang Talankar\_\_\_\_\_\_\_\_\_\_

ID:\_\_\_\_\_\_2020A7PS0083P\_\_\_\_\_\_\_\_\_\_\_Name:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Jaysheel Shah\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ID:\_\_\_\_\_\_2020A7PS0971P\_\_\_\_\_\_\_\_\_\_\_Name:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Deepam Desai\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ID:\_\_\_\_\_2020A7PS0033P\_\_\_\_\_\_\_\_\_\_\_\_Name:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Agrawal Rachit Mohit\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Mention the names of the Submitted files :

1\_\_\_\_\_lexer.c\_\_\_\_\_\_\_ 7\_\_\_\_\_\_driver.c\_\_\_\_\_\_\_\_ 13\_\_\_\_\_\_hash\_table.c\_\_\_\_

2\_\_\_\_\_lexer.h\_\_\_\_\_\_\_ 8\_\_\_\_\_\_stack.c\_\_\_\_\_\_\_\_\_ 14\_\_\_\_\_\_hash\_table.h\_\_\_\_

3\_\_\_\_\_lexerDef.h\_\_\_\_\_\_\_\_ 9\_\_\_\_\_stack.h\_\_\_\_\_\_\_\_\_ 15\_\_\_\_hash\_table\_def.h\_\_\_\_

4\_\_\_\_\_parser.c\_\_\_\_\_\_\_\_\_\_ 10\_\_\_\_stackDef.h\_\_\_\_\_\_ 16\_\_\_\_\_treeAdt.c\_\_\_\_\_\_\_\_

5\_\_\_\_\_parser.h\_\_\_\_\_\_\_\_\_ 11\_\_\_\_\_grammar.txt\_\_\_\_ 17\_\_\_\_\_treeAdt.h\_\_\_\_\_\_\_\_

6\_\_\_\_\_parserDef.h\_\_\_\_\_\_\_ 12\_\_\_\_tokenDef.h\_\_\_\_\_ 18\_\_\_\_\_treeAdtDEF.h\_\_\_

19 \_\_\_makefile\_\_\_\_\_\_\_\_\_ 20 \_\_\_\_\_testcase1.txt\_\_\_ 21 \_\_\_\_\_testcase2.txt\_\_\_\_\_\_\_\_\_

22 \_\_\_\_\_testcase3.txt\_\_ 23 \_\_\_\_\_testcase4.txt\_\_\_\_\_ 24 \_\_\_\_\_testcase5.txt\_\_\_\_\_\_\_\_\_

25 \_\_\_testcase6.txt\_\_ 26\_\_\_\_\_documentation.docx\_\_\_

1. Total number of submitted files: \_\_\_26\_\_\_\_\_\_\_\_ (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]
3. Have you compressed the folder as specified in the submission guidelines? (yes/no)\_\_\_\_\_Yes\_\_\_\_
4. **Lexer Details:**
   1. Technique used for pattern matching: By reading extra characters (if needed) from other classes and retracting (discarding) the input char accordingly.
   2. DFA implementation (State transition using switch case, graph, transition table, any other (specify):

Using switch case

* 1. Keyword Handling Technique: Searching in pre-filled hash-table(lookup table)
  2. Hash function description, if used for keyword handling: \_\_\_\_\_\_\_\_Based on ASCII value of the first character .Edge cases of underscore and AND, OR are handled.
  3. Have you used twin buffer? (yes/ no) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_yes\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
  4. Lexical error handling and reporting (yes/No):\_\_\_\_\_\_\_\_\_\_yes\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
  5. Describe the lexical errors handled by you: \_\_\_\_\_\_\_\_\_a) If length of identifier or keyword is greater than 20.\_\_\_\_b)Throws error if input character doesn’t have any transition to the next state in the DFA.
  6. Data Structure Description for tokenInfo (in maximum two lines):

tokenInfo is represented using the TOKEN data structure which has 3 fields: token (i.e. tokenName), value (i.e. the lexeme value), and line\_num (i.e. line number of the token). The value field is implemented using union for integer, real number and normal string respectively.

* 1. Interface with parser \_\_\_\_\_whenever parser calls for the getNextToken() function,the lexer which has the function returns the token.So,token is returned in demand by the parser.

1. **Parser Details:** 
   1. **High Level Data Structure Description (in maximum three lines each, avoid giving C definitions used):**
      1. grammar: We have used an array of doubly linked lists of the grammar struct type to store the grammar. The array stores multiple production rules of a non terminal as different elements of the array in order. The DLL stores the production rule for that non terminal.
      2. parse table: A 2-d array of int type of the size of no. of non-terminals X no of terminals. The parse table is initialized to -1. For a non -1 value, we are saving the row number from the grammar data structure. The index can be used to access the grammar rule for that non-terminal, terminal pair.
      3. parse tree: (Describe the node structure also) :The parse tree is the collection of treenodes which is a structure of token,first\_child,left\_sibling,right\_sibling ,parent and no of siblings.The parse tree is generated by our parsing function with help of getNextToken function and parsing stack.
      4. Parsing Stack node structure : It is a structure of a made sinf the structure of token and treenode. The token structure is used for getting the lexeme name and line number for parsing. “treenode” helps in the creation of the parse tree.

Any other (specify and describe) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

* 1. **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): We have used inorder traversal for printing the parse tree.
  2. **Grammar and Computation of First and Follow Sets** 
     1. Data structure for original grammar rules - An array of linked lists where every production rule in one node of the grammar
     2. FIRST and FOLLOW sets computation automated (yes /no)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_YES\_\_\_\_
     3. Data structure for representing sets: 2-D arrays of type int and size non-terminals X terminals. If a terminal is present in the first/follow set then the value of that tile is 1 else 0.
     4. Time complexity of computing FIRST sets \_\_[number of grammar rules]\*[number of terminals]\*[number of non terminals]\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
     5. Name the functions (if automated) for computation of First and Follow sets: first set computed using : first() function, and for follow: follow\_set().
     6. If computed First and Follow sets manually and represented in file/function (name that): NA
  3. **Error Handling** 
     1. Attempted (yes/ no):Yes
     2. Printing errors (All errors/ one at a time) :All the errors in the file are printed and the correct tokens are parsed.
     3. Describe the types of errors handled : Both syntactic and lexical errors have been handled in our code. The code traverses through the complete file printed the error and line no of error as and when faced.
     4. Synchronizing tokens for error recovery (describe): Follow set has been used as synchronizing tokens.
     5. Total number of errors detected in the given testcase t6(with\_syntax\_errors).txt

The code is able to detect a total of 12 syntactic errors and 2 lexical errors.

1. **Compilation Details:**
   1. Makefile works (yes/no):\_\_\_Yes\_\_\_
   2. Code Compiles (yes/ no): 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/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): Complete lexer and parser is running on all the six test cases provided. t1 can only be used for lexer. t2-t5 all are syntactically correct files. t-6 has syntactic and lexical errors that have been printed on the terminal by giving the error type and line number.
   2. Execution time taken for
      * t1.txt (in ticks) \_\_\_\_586\_\_\_\_\_\_\_\_\_\_\_\_\_ and (in seconds) \_\_\_\_0.000059\_\_\_\_\_
      * t2.txt (in ticks) \_\_\_\_190\_\_\_\_\_\_\_\_\_\_\_\_\_\_ and (in seconds) \_\_\_\_\_0.000019\_\_\_\_\_\_
      * t3.txt (in ticks) \_\_\_\_245\_\_\_\_\_\_\_\_\_\_\_\_\_\_ and (in seconds) \_\_\_\_\_0.0000245\_\_\_\_\_\_
      * t4.txt (in ticks) \_\_\_\_684\_\_\_\_\_\_\_\_\_\_\_\_\_\_ and (in seconds) \_\_\_\_\_0.0000684\_\_\_\_\_\_
      * t5.txt (in ticks) \_\_\_\_460\_\_\_\_\_\_\_\_\_\_\_\_\_\_ and (in seconds) \_\_\_\_\_0.000046\_\_\_\_\_\_
      * t6.txt (in ticks) \_\_\_\_1109\_\_\_\_\_\_\_\_\_\_\_\_\_ and (in seconds) \_\_\_\_\_0.0001109\_\_\_\_\_
   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
4. Specify the language features your lexer or parser is not able to handle (in maximum one line): The lexer and parser are able to handle all the language features given as per the specifications.
5. Are you availing the lifeline (Yes/No): \_\_\_\_\_Yes\_\_\_\_\_\_\_
6. Declaration: We, Aryan Desai, Ramakant Talankar, Jaysheel Shah, Deepam Desai, Rachit Agrawal 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:\_\_\_\_\_2020A7PS0123P\_\_\_\_\_\_\_\_\_\_\_\_Name:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Aryan Desai\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ID:\_\_\_\_\_2020A7PS0979\_\_\_\_\_\_\_\_\_\_\_\_\_Name:\_\_\_\_\_\_\_\_\_\_Ramakant Pandurang Talankar\_\_\_\_\_\_\_\_\_\_

ID:\_\_\_\_\_\_2020A7PS0083P\_\_\_\_\_\_\_\_\_\_\_Name:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Jaysheel Shah\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ID:\_\_\_\_\_\_2020A7PS0971P\_\_\_\_\_\_\_\_\_\_\_Name:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Deepam Desai\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ID:\_\_\_\_\_2020A7PS0033P\_\_\_\_\_\_\_\_\_\_\_\_Name:\_\_\_\_\_\_\_\_\_Agrawal Rachit Mohit\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Date: \_\_\_3rd March’ 2023\_\_\_\_\_

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

Should not exceed 4 pages.