**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. ID: 2017A7PS031P Name: Anirudh Goyal

ID:2017A7PS0069P Name: Anishkumar SS

ID:2017A7PS0122P Name: Rohit Jain

ID:2017A7PS0166P Name: Aditya Saxena

1. Mention the names of the Submitted files :

1. lexerDef.h 7. mappingTable.c 13. grammar.txt

2. lexer.h 8. parserDef.h 14. t1.txt

3. lexer.c 9. parser.h 15. t2.txt

4. Hash\_table.h 10. parser.c 16. t3.txt

5. driver.c 11. stack.c 17. t4.txt

6. mappingTable.h 12. arrayOfLinkedList.c 18. t5.txt

19. t6.txt 20. Makefile 21. custom\_test1.txt

22. custom\_test2.txt 23. custom\_test3.txt 24. coding details stage1.docx

1. Total number of submitted files: 23 (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
3. Have you compressed the folder as specified in the submission guidelines? (yes/no) Yes
4. **Lexer Details:**
   1. Technique used for pattern matching: DFA implementation with state transitions defined on each input. If an error is encountered the DFA goes to a trap state, reports an error and goes to the start state.
   2. DFA implementation (State transition using switch case, graph, transition table, any other (specify): State transition using switch case.
   3. Keyword Handling Technique: Search in hash table.
   4. Hash function description, if used for keyword handling:

If key is c1c2c3 where c1, c2, c3 are characters then the hash value is decimal equivalent of [bit(c1) bit(c2) bit(c3)] % (size of hash table) where bit(c) is the bit representation of the ASCII value of c.

* 1. Have you used twin buffer? Yes
  2. Lexical error handling and reporting? Yes
  3. Describe the lexical errors handled by you:

Identifier length more than 20 is an error and every trap state leads to an error.

* 1. Data Structure Description for tokenInfo (in maximum two lines):

Enum for token, line no., character array for lexeme, union to store numeric value of rnum or num, tag

* 1. Interface with parser : Passes one token at a time to the parser which is used to create the parse table when getNextToken() is invoked.

1. **Parser Details:** 
   1. **High Level Data Structure Description (in maximum three lines each, avoid giving C definitions used):**
      1. grammar : Array of linked lists in which each node stores the corresponding non terminal or terminal
      2. parse table : 2-d array. Each cell A[i][j] represents the rule number to be selected on the enumerated value of i as non-terminals and j as enumerated value of possible tokens. Syn(-2) and error(-1) entries are included.
      3. parse tree: (Describe the node structure also) : An n-ary tree. Each node stores information obtained from the lexer as TokenInfo, number of children and an array containing pointers to its children. The array space is dynamically allocated for each child.
      4. Parsing Stack node structure : Stack contains the pointers to the nodes of the parse tree. Basic stack operations like push, pop, etc. are defined accordingly.
      5. Hash table for keywords: contains a Data pair of key and value for storing keywords
      6. First and Follow Sets: Array of 64-bit integers of size = Number of non-terminals where each bit of an integer represents whether a given token is present in the set or not.
   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):

In-order traversal of the tree in which printing executes in the order: Leftmost child, root node, rest of the children. Printing format as specified is followed. 0 is printed for Epsilon as line number, $ as parent of Program.

* 1. **Grammar and Computation of First and Follow Sets** 
     1. Data structure for original grammar rules: Array of Linked Lists
     2. FIRST and FOLLOW sets computation automated (yes /no) : Yes
     3. Data structure for representing sets : Bit-set
     4. Time complexity of computing FIRST sets : O(n) where n is size of grammar
     5. Name the functions (if automated) for computation of First and Follow sets:

computeFirstAndFollow

calculateFirst

calculateFollow

populateFirstMap

populateFollowMap

* + 1. If computed First and Follow sets manually and represented in file/function (name that: NA
  1. **Error Handling** 
     1. Attempted (yes/ no): Yes
     2. Printing errors (All errors/ one at a time) : All errors
     3. Describe the types of errors handled:

All lexical errors

All syntax errors including token mismatch and panic mode recovery for non-terminals

* + 1. Synchronizing tokens for error recovery (describe) = Tokens in follow set are used as the synchronizing set for panic mode recovery. Integer value -2 is stored in the cell of parse table as SYN token.
    2. Total number of errors detected in the given testcase t6(with\_syntax\_errors).txt: 11

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: No
   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): All the functionalities are working as required successfully.
   2. Execution time taken for
      * t1.txt (in ticks) 819 and (in seconds) 0.000819
      * t2.txt (in ticks) 725 and (in seconds) 0.000725
      * t3.txt (in ticks) 1014 and (in seconds) 0.001014
      * t4.txt (in ticks) 1338 and (in seconds) 0.001338
      * t5.txt (in ticks) 1200 and (in seconds) 0.001200
      * t6.txt (in ticks) 1227 and (in seconds) 0.001227
   3. Gives segmentation fault with any of the test cases (1-6) uploaded on the course page. If yes, specify the testcase file name: None
4. Specify the language features your lexer or parser is not able to handle (in maximum one line): NA
5. Are you availing the lifeline (Yes/No): Yes
6. Declaration: We, Anirudh Goyal, Anishkumar SS, Rohit Jain, Aditya Saxena 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. [Write your ID and names below]

ID 2017A7PS031P Name: Anirudh Goyal

ID 2017A7PS0069P Name: Anishkumar SS

ID 2017A7PS0122P Name: Rohit Jain

ID 2017A7PS0166P Name: Aditya Saxena

Date: 24.2.20

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

Should not exceed 4 pages.