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

Batch No. :

**DEPARTMENT OF COMPUTER SCIENCE AND INFORMATION SYSTEMS**

**Compiler Construction (CS F363)**

**II Semester 2017-18**

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

**Coding Details**

**(February 26, 2018)**

1. **Personal details**

ID: 2015A7PS0549P

Name: Rajat Jain

1. **Files and folder details**
2. Mention the names of the Submitted files :

1. compiler.c 7. lexer.h 13. set.h 19. tree.h 34. codingDetails.docx

2. error.c 8. list.c 14. symbol.c 20. trie.c

3. error.h 9. list.h 15. symbol.h 21. trie.h

4. grammar.c 10. parser.c 16. token.c 22. makefile

5. grammar.h 11. parser.h 17. token.h 23. grammar.txt

6. lexer.c 12. set.c 18. tree.c 24. testcases 1-10.txt

1. Total number of submitted files: 34 (All files should be in ONE folder named exactly as your ID)
2. Have you compressed the folder as specified in the submission guidelines? Yes
3. **Lexer Details:**
   1. Technique used for pattern matching:  
       A hard-coded DFA that reads a lookahead from the buffer and tokenizes input based on final states. If a final state is not reached, an error is reported.
   2. Keyword Handling Technique:   
       A trie stores mappings of keywords to token type.
   3. Hash function description, if used for keyword handling:

Nil.

* 1. Have you used twin buffer? (yes/ no) :  
      Yes. See lexer.c commenting on description of the implementation.
  2. Error handling and reporting (yes/ no):  
      Yes
  3. Describe the errors handled by you   
      1. File Not Opening: Raised when the source file cannot be opened.  
      2. Symbol Not Recognized: If a character encountered is not in language spec.  
      3. Pattern Not Recognized: If a pattern encountered is not in language spec.  
      4. Identifier Size Exceeded: If an identifier exceeds 20 characters.  
      5. String Size Exceeded: if a string exceeds 20 characters.
  4. Data Structure Description for tokenInfo (in maximum two lines):   
      TokenInfo contains token type (enum), line number (integer) and value (union of lexeme, integer, real and string which is populated according to type).

1. **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 non terminals which in turn contain their symbol type, a list of their rules (List of lists), a list of their occurrences (for follow set computation) and their first and follow sets. EPSILON in rules is denoted by ‘$’ and max number of NTs is 50.
      2. parse table:   
         Parse table is a 2 dimensional array of size (No. of Non Terminals)X(No. of terminals) containing pointers to rules. Null rules are represented by linked lists of length zero.
      3. parse tree: (Describe the node structure also)  
         Parse tree is an n-ary tree where each node contains a symbol structure, a pointer to a list of its children and a pointer to its parent. The symbol structure in turn contains information on symbol type, whether it is a terminal and a token ptr in case of terminal.
   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. If a node doesn’t have children, its info is printed, else the leftmost child is printed first, followed by the parent and then the remaining.
   3. Computation of First and Follow Sets
      1. Data structure for First and Follow sets:  
         Bit vector representation of sets are used with uint64\_t as the underlying datatype. A bit is 0 if element does not belong to the list, else it is 1. EPSILON is represented by including ‘63’ in the set.
      2. FIRST and FOLLOW sets computation automated (yes /no): Yes
      3. Name the functions (if automated) for computation of First and Follow sets:  
         1. createFirstSets: Creates first set for the entire grammar by calling first.  
         2. createFollowSets: Creates first set for the entire grammar by calling follow.
      4. If computed First and Follow sets manually and represented in file/function (name that): NA
   4. Error Handling and recovery
      1. Attempted (yes/ no): Yes
      2. Synchronizing set formation details:  
         Each synchronizing set contains the first and follow set of the non-terminal along with SEMICOLON.
      3. Describe the types of errors handled:  
         1. Unexpected Symbol: If a syntactic error arises due to unexpected token.  
         2. Unexpected Termination: If token stream terminates but stack is non empty.  
         3. Longer Token Stream: If the stack is empty but token stream is not.
2. **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
3. **Driver Details:** Does it take care of the options specified earlier(yes/no): Yes
4. **Execution details**
   1. Status (describe in maximum 2 lines): Executes as expected.
   2. Gives segmentation fault with any of the revised test cases (1-5) uploaded on the course page. If yes, specify the testcase file name: No
5. Specify the language features your lexer or parser is not able to handle (in maximum one line): All features are handeled.
6. **Lifeline detail:** Are you availing the lifeline (Yes/No): No
7. **Declaration**: I, Rajat Jain (your name) declare that I have put my genuine efforts in creating the compiler project code and have submitted the code developed only by me. I have not copied any piece of code from any source. If my code is found plagiarized in any form or degree, I understand that a disciplinary action as per the institute rules will be taken against me and I will accept the penalty as decided by the department of Computer Science and Information Systems, BITS, Pilani.

ID: 2015A7PS0549P

Name: Rajat Jain

Date: 25-2-2018

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

/\*not to exceed two pages\*/