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

**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: **2015A7PS0088P**

Name: **PARTHO SARTHI**

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

**1 driver.c 7 parserDef.c 13 firstAndFollow.c 19 parseTable.csv**

**2 lexer.c 8 parserDef.h 14 firstAndFollow.h 20 testcase1.txt**

**3 lexer.h 9 symbolDef.c 15 makefile 21 testcase2.txt**

**4 langDef.h 10 symbolDef.h 16 grammar.txt 22 testcase3.txt**

**5 parser.c 11 symbolStack.c 17 first.txt 23 testcase4.txt**

**6 parser.h 12 symbolStack.h 18 follow.txt 24 testcase5.txt**

1. Total number of submitted files: **24** (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/no) **YES**
3. **Lexer Details:**
   1. Technique used for pattern matching: **The DFA ‘getToken()’ is implemented using nested switch cases for current state and current input character. This function returns a <TokenInfo> with all the info about token.**
   2. Keyword Handling Technique: **Using multiple string comparison statements.**
   3. Hash function description, if used for keyword handling: **N.A.**
   4. Have you used twin buffer? (yes/ no): **NO**
   5. Error handling and reporting (yes/No): **YES**
   6. Describe the errors handled by you : **1) Identifier is longer than the prescribed length. 2) Function Identifier is longer than the prescribed length. 3) String is longer than the prescribed length. 4) Unknown Symbol found 5) Unknown Pattern found.**
   7. Data Structure Description for tokenInfo (in maximum two lines): <**TokenInfo.id> contains the id for the token(which is mapped to an actual token value). <TokenInfo.name> contains the actual value of the lexeme. <TokenInfo.line> line number where the token occurs.**
4. **Parser Details:** 
   1. High Level Data Structure Description (in maximum three lines each, avoid giving C definitions used):
      1. Grammar :  **Every (non-terminal,terminal) is stored as a <SymbolDef> Data Structure. Every symbol has <RuleList> which is linked list of <Rule> for that symbol. <Rule> is a linked list of <SymbolDef> present in the right hand of grammar for that symbol. Thus all the symbols are internally connected creating a Graph of symbols and highly efficient in terms of space and time.**
      2. Parse table: **It is a 2-D array consisting of symbol pointer <SymbolDef> and the rule no <int>.**
      3. Parse tree: (Describe the node structure also): **It is a n-ary Tree having node as <SymbolDef>,firstChild,parent,sibling. Each child points to the next sibling. The terminals do not have any child.**
      4. Symbol: **As described earlier, <SymbolDef> stores the value<character array>, isTerminal<int>,rules <RuleList>,first<SymbolList>,follow<SymolList>.**
      5. SymbolList: **It is a linked list of <SymbolDef>. It is used at several places e.g to store the synchronization set,first and follow of a symbol, rules.**
      6. Symbol Stack: **It is a stack of <SymbolDef>. It is used by parser to verify if the source code is syntactically correct 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):  
         **The tree is traversed inorder and the nodes are printed accordingly. The format is:  
         ("Lexeme","Line","Token","Value If Number","Parent Node","Leaf","Node")**
   3. Computation of First and Follow Sets
      1. Data structure for First and Follow sets: **Stored as <SymbolList> first and <SymbolList> follow for every <SymbolDef>**
      2. FIRST and FOLLOW sets computation automated (yes /no): **YES**
      3. Name the functions (if automated) for computation of First and Follow sets: **findFirstAndFollow()**
      4. If computed First and Follow sets manually and represented in file/function (name that): **N.A. (Although after automated computation the sets are saved as first.txt and follow.txt for the user)**
   4. Error Handling and recovery
      1. Attempted (yes/ no): **YES**
      2. Synchronizing set formation details: **The synchronizing set consists of the FOLLOW(X) and SEMICOLON**
      3. Describe the types of errors handled: **Apart from Lexical Errors(handled by the lexer) two types of syntactical errors occur-** 
         1. **Terminal at top of stack do not match with the input token** 
            1. **Both the input pointer and the stack symbol are popped.**
         2. **Parse Table entry corresponding to the Non Terminal and input token do not match** 
            1. **Panic Mode Recovery is used to recover. The input token is skipped till it is not present in the synchronizing set of the Non Terminal. After that the Non Terminal is popped and normal state is restored.**
5. **Compilation Details**
   1. Makefile works (yes/no): **YES**
   2. Code Compiles (yes/ no): **YES**
   3. Mention the .c files that do not compile: **NIL**
   4. Any specific function that does not compile: **NIL**
   5. Ensured the compatibility of your code with the specified gcc version(yes/no) **YES**
6. **Driver Details:** Does it take care of the options specified earlier(yes/no): **YES**
7. **Execution details**
   1. status (describe in maximum 2 lines): **All modules are implemented and works correctly with all the testcases provided.**
   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**
8. Specify the language features your lexer or parser is not able to handle (in maximum one line): **NIL**
9. **Lifeline detail:** Are you availing the lifeline (Yes/No): **NO**
10. **Declaration**: I, PARTHO SARTHI, 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: **2015A7PS0088P**

Name: **PARTHO SARTHI**

Date: **26/02/2018**