**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-2 Submission)**

**Coding Details**

**(April 20, 2018)**

*Instruction: Write the details precisely and neatly. Places where you do not have anything to mention, please write NA for Not Applicable.*

1. ID Number: 2015A7PS0140P

Name: TANVI AGGARWAL

1. Mention the names of the Submitted files ( Include Stage-1 and Stage-2 both)

1\_\_\_\_parser.c\_\_\_\_\_ 10\_\_\_\_driver.c\_\_\_\_\_\_\_\_ 19\_\_\_\_testcase4.txt\_\_\_\_\_\_\_

2\_\_\_\_parser.h\_\_\_\_\_\_ 11\_\_\_\_symbolTable.c\_\_ 20\_\_\_\_testcase5.txt\_\_\_\_\_\_\_

3\_\_\_\_parserDef.h\_\_\_ 12\_\_\_\_Ffirsts.txt\_\_\_\_\_\_\_ 21\_\_\_\_codingDetails.docx\_

4\_\_\_\_lexer.c\_\_\_\_\_\_\_\_ 13\_\_\_\_Ffollows.txt\_\_\_\_\_ 22\_\_\_\_makefile\_\_\_\_\_\_\_\_\_\_\_

5\_\_\_\_lexer.h\_\_\_\_\_\_\_\_ 14\_\_\_\_Fgrammar.txt\_\_\_ 23\_\_\_\_typeChecker.h\_\_\_\_\_

6\_\_\_\_lexerDef.h\_\_\_\_\_ 15\_\_\_\_testcase1.txt\_\_\_\_\_\_ 24\_\_\_\_typeChecker.c\_\_\_\_\_

7\_\_\_\_ast.h\_\_\_\_\_\_\_\_\_\_\_ 16\_\_\_\_testcase2.txt\_\_\_\_\_\_ 25\_\_\_\_semantic.h\_\_\_\_\_\_\_\_

8\_\_\_\_ast.c\_\_\_\_\_\_\_\_\_\_\_ 17\_\_\_\_testcase3.txt\_\_\_\_\_\_ 26\_\_\_\_semantic.c\_\_\_\_\_\_\_\_

9\_\_\_\_symbolTable.h\_\_ 18\_\_\_\_testcase6.txt\_\_\_\_\_\_

1. Total number of submitted files: \_\_\_\_26\_\_\_\_\_ (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. **Status of Code development**: Mention 'Yes' if you have developed the code for the given module, else mention 'No'.
   1. Lexer (Yes/No): \_\_\_\_\_\_\_\_yes\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
   2. Parser (Yes/No):\_\_\_\_\_\_\_\_yes\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
   3. Abstract Syntax tree (Yes/No):\_\_\_yes\_\_\_\_\_\_\_\_\_\_\_\_
   4. Symbol Table (Yes/ No):\_\_\_\_\_\_\_\_yes\_\_\_\_\_\_\_\_\_\_\_\_
   5. Type checking Module (Yes/No):\_\_\_\_\_\_partially\_\_\_\_\_\_\_
   6. Semantic Analysis Module (Yes/ no):\_\_\_\_\_partially\_\_\_\_\_\_\_(reached LEVEL \_\_2\_ as per the details uploaded)
   7. Code Generator (Yes/No):\_\_\_\_\_\_\_no\_\_\_\_\_\_\_\_\_\_
4. **Execution Status**:
   1. Code generator produces code.asm (Yes/ No):\_\_\_\_\_no\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
   2. code.asm produces correct output using NASM for testcases (C#.txt, #:1-3):\_\_\_\_\_\_\_\_\_NA\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
   3. Semantic Analyzer produces semantic errors appropriately (Yes/No):yes (partially)
   4. Type Checker reports type mismatch errors appropriately (Yes/ No): yes (partially)
   5. Symbol Table is constructed (yes/no)\_\_\_\_yes\_\_\_\_\_and printed appropriately (Yes /No):\_\_\_\_\_\_\_\_yes\_\_\_\_\_\_\_\_
   6. AST is constructed (yes/ no) \_\_\_\_\_\_yes\_\_\_\_\_\_and printed (yes/no) \_\_\_\_yes\_\_\_\_\_\_
   7. Name the test cases out of 9 as uploaded on the course website for which you get the segmentation fault (testcase#.txt ; # 1-6 and c@.txt ; @:1-3):\_\_\_\_\_\_\_\_\_\_\_NA\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
5. **Data Structures** (Describe in maximum 2 lines and avoid giving C definition of it)
   1. AST node structure: Each node contains – integer corresponding to ruleUsed of grammar; lexeme; node type (ID); lineNo; pointers to firstChild, parent and sibling.
   2. Symbol Table structure: Each node corresponding to a symbolTable stores the following information – scope(function\_name); hashTable (which stores entries which are of the form [name, type, width, offset]); nesting\_level; pointers to firstChild, sibling and parent of that node.
   3. Matrix type expression structure:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
   4. Input parameters type structure:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
   5. Output parameters type structure:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
   6. Structure for maintaining the three address code(if created) :\_\_\_\_\_\_\_\_\_\_\_NA\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

1. **Semantic Checks:** Mention your scheme NEATLY for testing the following major checks (in not more than 5-10 words)[ Hint: You can use simple phrases such as 'symbol table entry empty', 'symbol table entry already found populated', 'traversal of linked list of parameters and respective types' etc.]
   1. Variable not Declared : symbol table entry empty
   2. Multiple declarations: symbol table entry already found populated
   3. Number and type of input and output parameters:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
   4. assignment of value to the output parameter in a function \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
   5. function call semantics:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
   6. type checking :\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
   7. return semantics:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
   8. Recursion :
   9. module overloading: function\_name matches with another one in parent’s scope
   10. 'If' semantics :\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
   11. Matrix semantics and type checking of matrix type variables: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

* 1. register allocation (your manually selected heuristic) :\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
  2. Scope of variables and their visibility : symbol table entry should be present (otherwise keep checking in symbol table of ancestors of node).

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. Specify the language features your compiler is not able to handle (in maximum one line)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_NA\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Are you availing the lifeline (Yes/No): \_\_\_\_No\_\_\_\_\_\_\_\_\_\_
2. Write exact command you expect to be used for executing the code.asm using NASM simulator [We will use these directly while evaluating your NASM created code]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_NA\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

1. **Strength of your code**(Strike off where not applicable): (a) correctness (b) ~~completeness~~ (c) robustness (d) Well documented (e) readable (f) strong data structure (f) Good programming style (indentation, avoidance of goto stmts etc) (g) modular (h) space and time efficient
2. Any other point you wish to mention: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

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

1. **Declaration:** I, Tanvi Aggarwal declare that I have put my genuine efforts in creating the compiler project code and have submitted the code developed 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.

Sign:\_\_\_\_Tanvi\_\_

ID: 2015A7PS0140P

Name: TANVI AGGARWAL

Date: 20/04/2018

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

/\*not to exceed three pages\*/