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

**Group No.**

**26**

Batch No. :

**DEPARTMENT OF COMPUTER SCIENCE AND INFORMATION SYSTEMS**

**Compiler Construction (CS F363)**

**II Semester 2018-19**

**Compiler Project (Stage-2 Submission)**

**Coding Details**

**(April 14, 2019)**

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

1. IDs and Names of team members

ID: 2016A7PS0014P Name: Nithin Benny Myppan

ID: 2016A7PS0028P Name: Adhitya Mamallan

ID: 2016A7PS0080P Name: Swarup N

ID: 2016A7PS0111P Name: Naveen Unnikrishnan

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

1 lexer.c 7 NaryTree.c 13 ast.c 19 driver.c

2 lexer.h 8 NaryTree.h 14 ast.h 20 makefile

3 lexerDef.h 9 lookup.c 15 astDef.h 21 grammar.txt

4 parser.c 10 lookup.h 16 symbolTable.c

5 parser.h 11 stack.c 17 symbolTable.h

6 parserDef.h 12 stack.h 18 symbolTableDef.h

1. Total number of submitted files: 21 (All files should be in ONE folder named exactly as **Group\_26**, **26** is our group number)
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): **No**
   6. Semantic Analysis Module (Yes/ no): **Yes** (reached LEVEL **2** as per the details uploaded)
   7. Code Generator (Yes/No): **Yes**
4. Execution Status:
   1. Code generator produces code.asm (Yes/ No): **No**
   2. code.asm produces correct output using NASM for testcases (Main#.txt, #:1-4: **No**
   3. Semantic Analyzer produces semantic errors appropriately (Yes/No): **No**
   4. Type Checker reports type mismatch errors appropriately (Yes/ No): **No**
   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 7 as uploaded on the course website for which you get the segmentation fault (testcase#.txt ; # 1-3 and Main@.txt ; @:1-4): **NA**
5. Data Structures (Describe in maximum 2 lines and avoid giving C definition of it)
   1. AST node structure: Tree where nodes have pointers to linked list of children, and attributes like label, pointer to parent, next node on same level, scope, token type, calling function, lexical unit, line number and value.
   2. Symbol Table structure: Linked lists where nodes have pointers to lexical unit, datatype, offset, width, AST node, value and optional record definition
   3. Data structure for global variables: Entries in the symbol table
   4. Record type expression structure: Linked list of individual fields, with number of fields, width and name
   5. Input parameters type structure: Passed by reference
   6. Output parameters type structure: Pointers, some outputs are taken in input buffers
   7. Structure for maintaining the three address code(if created) : NA
   8. Any other interesting data structure used: NA
6. Semantic Checks: Mention your scheme NEATLY for testing the following major checks
   1. Variable not Declared: The AST node’s name is not found in the symbol table, when its token is TK\_ID
   2. Multiple declarations: The AST’s node name is already found in the symbol table, when its parent’s label is TK\_DECLARATION
   3. Number and type of input and output parameters: NA
   4. assignment of value to the output parameter in a function: NA
   5. function call semantics: NA
   6. type checking: NA
   7. return semantics: NA
   8. Recursion : AST node’s lexeme is same as AST node’s calling fn’s lexeme
   9. function overloading: The AST’s node name is already found in the symbol table, when its label is TK\_FUNID and its parent’s label is FN
   10. 'while' loop semantics : NA
   11. record data type semantics and type checking of record type variables : NA
   12. register allocation: NA
   13. Scope of variables and their visibility : NA
7. 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**
8. Driver Details: Does it take care of the options specified earlier?(yes/no): **Yes**
9. Specify the language features your compiler is not able to handle (in maximum one line)

Type checking, semantic analysis (limited), intermediate code generation, assembly code generation

1. Are you availing the lifeline (Yes/No): **Yes**
2. Write exact command you expect to be used for executing the code.asm using NASM simulator:

**NA**

1. Strength of your code(Tick the boxes where applicable): (a) correctness ☑ (b) completeness (c) robust (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: **NA**
3. Declaration: We, **Nithin Benny Myppan, Adhitya Mamallan, Swarup N, and Naveen Unnikrishnan** declare that we have put our genuine efforts in creating the compiler project code and have submitted the code developed by us. 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.

Date: **15/04/2019**