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

Batch No. : 33

**Group No.33**

**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:\_\_\_\_\_\_\_\_2016A7PS0036P\_\_\_\_\_\_\_\_\_\_\_\_Name:\_\_\_\_\_\_\_\_\_\_Megh\_Thakkar\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ID:\_\_\_\_\_\_\_\_2016A7PS0103P\_\_\_\_\_\_\_\_\_\_\_\_Name:\_\_\_\_\_\_\_\_\_\_Sahil\_Singla\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ID:\_\_\_\_\_\_\_\_2016A7PS0110P\_\_\_\_\_\_\_\_\_\_\_\_Name:\_\_\_\_\_\_\_\_\_\_Sankalp\_Sangle\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ID:\_\_\_\_\_\_\_\_2016A7PS0150P\_\_\_\_\_\_\_\_\_\_\_\_Name:\_\_\_\_\_\_\_\_\_\_Patel\_Parth\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

1. driver.c 2. ast.c 3. ast.h 4. astDef.h 5. hashTable.c 6.hashTable.h

7. hashTableDef.h 8. lexer.c 9. lexer.h 10. lexerDef.h 11. NaryTree.c

12. NaryTree.h 13. NaryTreeDef.h 14. parser.c 15. parser.h 16. parsesrDef.h

17. stack.c 18. stack.h 19. stackDef.h 20. symbolTable.c 21. symbolTable.h

22. symbolTableDef.h 23. typeChecker.c 24. typeChecker.h

1. Total number of submitted files: \_\_\_\_24\_\_\_\_\_\_\_(All files should be in ONE folder named exactly as Group\_#, # is your 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):\_\_\_\_\_\_Yes\_\_\_\_\_\_\_\_\_\_\_\_\_
   6. Semantic Analysis Module (Yes/ no):\_\_\_\_Yes\_\_\_\_\_(reached LEVEL \_\_4\_\_ as per the details uploaded)
   7. Code Generator (Yes/No):\_\_\_\_\_\_No\_\_\_\_\_\_\_\_\_\_\_
4. Execution Status:
   1. Code generator produces code.asm (Yes/ No):\_\_\_\_\_\_\_\_\_Yes\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
   2. code.asm produces correct output using NASM for testcases (Main#.txt, #:1-4): \_\_\_\_\_\_\_No\_\_\_\_\_\_\_\_\_\_\_\_
   3. Semantic Analyzer produces semantic errors appropriately (Yes/No):\_\_\_\_Yes\_\_\_\_\_\_\_\_
   4. Type Checker reports type mismatch errors appropriately (Yes/ No):\_\_\_\_\_Yes\_\_\_\_\_\_\_\_
   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:\_\_\_\_\_\_\_\_\_\_\_\_\_\_AST Node has lexical token attributes and has a label attribute to refer to its type(for example INTNODE, REALNODE etc)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

* 1. Symbol Table structure:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_hashTable using Horner’s theorem. Entries correspond to symbol table nodes and contain details in accordance with AST nodes.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

* 1. Data structure for global variables: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_is\_global attribute in the symbol table nodes set when a GLOBALNODE is found while traversing the AST, searching for identifiers.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
  2. Record type expression structure:\_\_\_\_\_\_\_\_\_\_\_\_\_\_The structure has a further linked list which stores tokens as per the attributes defined by the program and encountered while traversing the AST.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
  3. Input parameters type structure:\_\_\_\_\_\_Linked list of individual data structures storing necessary information about the function(similar to the lexical token data structure) and attached with the function node in the function table.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
  4. Output parameters type structure:\_\_\_\_Linked list of individual data structures storing necessary information about the function(similar to the lexical token data structure) and attached with the function node in the function table.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
  5. Structure for maintaining the three address code(if created) :\_\_\_\_NA\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
  6. Any other interesting data structure used \_\_\_\_\_A record table which iterates through individual record fields which are helpful when prompted to print in the required format efficiently(for example, “,” separated files)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

1. Semantic Checks: Mention your scheme NEATLY for testing the following major checks
   1. Variable not Declared :\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_retrieve function in symbol table\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
   2. Multiple declarations: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_retrieve function in symbol table when encountering new declaration\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
   3. Number and type of input and output parameters:\_\_\_\_\_Verified in the input parameters and output parameters linked list stored in the function table\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
   4. assignment of value to the output parameter in a function \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_is\_assigned attribute in symbol table node, updated when ASSIGNOP\_NODE encountered for the identifier when traversing the AST\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
   5. function call semantics:\_ Using simultaneous searches in the function table(for example, to prevent recursive calls) and traversal of AST to verifiy the type matching of return parameters.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
   6. type checking\_\_\_Identifiers involved in arithmetic operations are retrieved from the symbol tables where their types are stored. Simple if-else conditions are further employed whenever it is required to verify types.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
   7. return semantics:\_\_\_\_\_\_type checking is done when traversing the AST to verify inputs to the return array match with the definitions as well as cross verification with the defined output parameters stored in the function table\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
   8. Recursion :\_\_\_\_\_\_\_\_\_\_\_\_\_\_Whenever a function call is encountered, the parent function is examined according to the AST. If they match, it is a recursive call.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
   9. function overloading:\_\_\_\_\_By verifying the input parameters in a function table with the input parameter linked list stored originally in the function table.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
   10. 'while' loop semantics : \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Verifying that the identifier involved in loop test conditions has been assigned a new value in the loop body (tentatively to update the value such that it approaches the termination condition)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
   11. record data type semantics and type checking of record type variables : \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Records stored in the Record table can be used for verifying the type variables as they contain a linked list of attributes encountered during record declaration.\_The same can be used for type semantics as well.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

* 1. register allocation:\_\_\_NA\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
  2. Scope of variables and their visibility :\_\_\_\_\_\_\_\_\_\_\_\_Each variable declared has a pointer to the function node within which it was declared, if a GLOBALNODE is not encountered just after it. Variables having a common function node hence possess the same scope.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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): \_\_\_\_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: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

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

1. Declaration: We, \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Megh Thakkar, Sankalp Sangle, Sahil Singla \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ and \_Parth Patel\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (your names) 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:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Not to exceed beyond 3 pages)