**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: 2015A7PS0549P

Name: Rajat Jain

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

1. compiler.c 7. lexer.h 13. set.h 19. tree.h 25. ast.h

2. error.c 8. list.c 14. symbol.c 20. trie.c 26. codegenerator.c

3. error.h 9. list.h 15. symbol.h 21. trie.h 27. codegenerator.h

4. grammar.c 10. parser.c 16. token.c 22. makefile 28. intermediate.c

5. grammar.h 11. parser.h 17. token.h 23. grammar.txt 29. intermediate.h

6. lexer.c 12. set.c 18. tree.c 24. ast.c 30. quadruple.c

31. quadruple.h 33. semantic.h 35. symboltable.h 37. type.h 39. testcase1-13.txt

32. semantic.c 34. symboltable.c 36. type.c 38. functions.asm 41. c1-10.txt

42. codingDetails2.docx

1. Total number of submitted files: 62 (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. **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): Yes
4. **Execution Status**:
   1. Code generator produces code.asm (Yes/ No): Yes
   2. code.asm produces correct output using NASM for testcases (C#.txt, #:1-3): Yes
   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 9 as uploaded on the course website for which you get the segmentation fault (testcase#.txt ; # 1-6 and c@.txt ; @:1-3): None
5. **Data Structures** (Describe in maximum 2 lines and avoid giving C definition of it)
   1. AST node structure: Each tree has a root symbol, a parent pointer and 3 attributes which are used for AST generation, type checking and code generation.
   2. Symbol Table structure: A hashtable that contains 2 types of entries: ID and FUNID.  
      ID Type contains: Tree reference, Size, Offset, Type, isDefined  
      FUNID Type contains: Tree reference, Scope, I/P and O/P parameters.
   3. Matrix type expression structure: A non-terminal node which has 1 child for every row in the matrix which in turn have integer elements as its children.
   4. Input parameters type structure: A non-terminal node which has 1 child-tree of the form type→identifier for each I/P parameter. (Type is the parent of identifier)
   5. Output parameters type structure: Same as above
   6. Structure for maintaining the three address code(if created) : A quadruple with operator type and 3 address types which in turn can contain constants and symbol table entries.
6. **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 not found
   2. Multiple declarations:   
      Symbol Table entry already existed while trying to insert a new ID
   3. Number and type of input and output parameters:  
      Fetch parameter lists from symbol table and compare input params with function call and output params with variables on right side of ASSIGNOP
   4. assignment of value to the output parameter in a function:  
      Mark the variable as defined in the symbol table and check if all o/p parameters are defined while finishing function definition.
   5. function call semantics: It is allowed in arithmetic expressions if only 1 parameter is returned. It is allowed as a statement if no parameters are returned. All calls also check passed parameter types with I/p list by iterating over both of them simultaneously.
   6. type checking: Assign the operator a type according to the type checking rules mentioned and compute the expression type in a bottom up fashion. IF an error occurs, assign error-type to the operator and report it.
   7. return semantics: Iterate over output parameter list and check if all the variables have been assigned a value using their symbol table entries.
   8. Recursion: Fetch the FUNID tree reference from symbol table and compare with tree references of all the parent functions of the current scope, if they have same address in the tree, it is a recursive call.
   9. module overloading: If the FUNID entry is present in the symbol table while declaring.
   10. 'If' semantics: Check if boolean expression returns a boolean type by type-checking the boolean expression.
   11. Matrix semantics and type checking of matrix type variables:   
       Check if matrix is rectangular by iterating over all children and checking whether the number of children in each row are equal. If any dimension is >10, report error.
   12. register allocation (your manually selected heuristic): ebp is used as base pointer, esp is used as stack pointer, dl is used as a counter and eax is used to pass arguments to asm functions. All templates are atomic, so register allocation is hard-coded.
   13. Scope of variables and their visibility: A variable is visible if its entry is present in the current symbol table or its ancestor symbol tables which can be accessed by st->parent
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)

Functions and real do not generate any code.

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 [We will use these directly while evaluating your NASM created code]

$ nasm -f elf code.asm

$ ld -m elf\_i386 code.o -o code

$ ./code

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, Rajat Jain (your name) 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:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ID: 2015A7PS0549P

Name: Rajat Jain

Date: 20-04-18

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

/\*not to exceed three pages\*/