**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: **2015A7PS0076P**

Name: **Shivankit Gaind**

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

1 codeGen.h 11 lookUpTable.h 21 Stack.c

2 optimizer.h 12 lexer.h 22 NaryTree.c

3 intermediateCode.h 13 codeGen.c 23 lookUpTable.c

4 typeExtractor.h 14 optimizer.c 24 lexer.c

5 semanticAnalyzer.h 15 intermediateCode.c 25 intermediateCodeDef.h

6 symbolTable.h 16 typeExtractor.c 26 typeExtractorDef.h

7 ast.h 17 semanticAnalyzer.c 27 semanticAnalyzerDef.h

8 parser.h 18 symbolTable.c 28 symbolTableDef.h

9 Stack.h 19 ast.c 29 astDef.h

10 NaryTree.h 20 parser.c 30 parserDef.h

31 StackDef.h 32 NaryTreeDef.h 33 lookUpTableDef.h

34 lexerDef.h 35 makefile 36 driver.c

37 grammar.txt 38 printInteger.asm 39 testcase(1-6).txt – 6 files

40 c(1-3).txt – 3 files 41 codingDetails.docx

1. Total number of submitted files: **48** (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): **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): **All working fine**
5. **Data Structures** (Describe in maximum 2 lines and avoid giving C definition of it)
   1. AST node structure: Each Node has a pointer to a linked list of children, it’s own parent, it’s next sibling in the linked list, a ‘concat’ pointer for next sibling in the compressed list attatched only to it, and a pointer to a Symbol Table Tree node corresponding to it’s current scope. It also has pointer to lexical token(for leaf nodes only) and dimensions parameter for matrix nodes.
   2. Symbol Table structure: An Nary Tree(of hashtables), with each tree node representing the scope of a function. Each tree node has a symbol table which is implemented as a hashtable.
   3. Matrix type expression structure: Compressed List of ‘ROW’ nodes and each row has compressed list of ‘NUM’ nodes; Dimension’s parameter in AST Node for whole matrix.
   4. Input parameters type structure: Compressed List of AST Nodes with label PARAMETER\_NODE as well as a datatype field.
   5. Output parameters type structure: Compressed List of AST Nodes with label PARAMETER\_NODE as well as a datatype field.
   6. Structure for maintaining the three address code(if created) : A Linked List of Quadruples(each with fields corresponding to operator, arg1, arg2 and result).
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 : No symbol Table entry in current/parental hierarchy scope.
   2. Multiple declarations: Symbol Table entry found either in current scope or one of the parent’s scope.
   3. Number and type of input and output parameters: First find the AST node of Function Definition corresponding to function call from the symbol table (since each symbol has a pointer to the corresponding AST Node)and then match the number and type of actual and formal parameters using traversal of linked lists.
   4. assignment of value to the output parameter in a function: An ‘assigned’ variable is associated with every symbol in symbol table. If it’s 0, then error for that parameter.
   5. function call semantics: AST node of Function Definition corresponding to function call is found from the symbol table. Then datatypes of actual parameters are matched with those of formal parameters by traversing both lists parallely.
   6. type checking : Each AST Node has a nodeType parameter, which is used to check whether operands of an operator or LHS and RHS of assignment statement match properly.
   7. return semantics: AST node has linked list of datatypes for return parameters, and those are matched with the datatypes of receiving parameters. AST node of Function Definition corresponding to function call is found from the symbol table.
   8. Recursion : Each node has pointer to parent FUNID node, check if the lexemes match, then it’s a recursive call.
   9. module overloading: Check if symbol table has entry for the same module name in the current scope.
   10. 'If' semantics : Do Type Extraction and Type Checking of Corresponding Boolean Expression and then perform semantic analysis on statements inside the if/else blocks.
   11. Matrix semantics and type checking of matrix type variables: AST node maintains a nodeType parameter as well as ‘dimensions’ parameter for MATRIX\_NODE labeled nodes, and hence type checking can be done using it.
   12. register allocation (your manually selected heuristic) : Round Robin Fashion, code is such that at a time, no more than 3 registers are in use.
   13. Scope of variables and their visibility : Using presence/absence in Symbol Table (corresponding to current scope/parent scopes)
7. **Compilation Details**:
   1. Makefile works (yes/No): **Yes**
   2. Code Compiles (Yes/ No): **Yes**
   3. Mention the .c files that do not compile: **All files compile properly**
   4. Any specific function that does not compile: **All functions compile properly**
   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)

In code Generation Part, it cannot handle read() statement.

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**

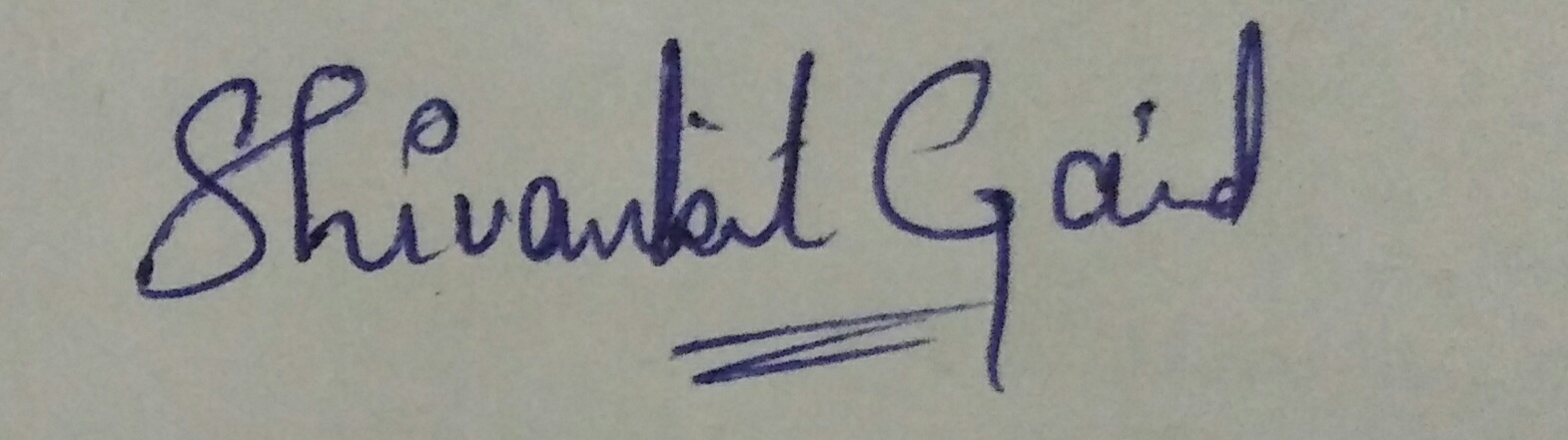
**ld -m elf\_i386 code.o -o executable**

**./executable**

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

**All features are there in the code with good strength.**

1. Any other point you wish to mention: My code can also handle errors like ‘Uninitialized Variable used on RHS of an Assignment Statement’ and also does some static computations as optimization.
2. **Declaration:** I, **Shivankit Gaind** (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: On Left

ID: **2015A7PS0076P**

Name: **Shivankit Gaind**

Date: **21 April, 2018**

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

/\*not to exceed three pages\*/