BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI
DEPARTMENT OF COMPUTER SCIENCE AND INFORMATION SYSTEMS

Group Number
15

Compiler Construction (CS F363)
II Semester 2021-22
Compiler Project (Stage-2 Submission)
Coding Details
(April 16, 2022)

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:__2019A7PS0088P_____Name:__Preetika Verma_____

   ID:__2019A7PS1140P_____Name:__Pritika Ramu_____

   ID:__2019A7PS0097P_____Name:__Nandan Parikh_____

   ID:__2019A7PS0042P_____Name:__Sneha_____

   ID:__2019A7PS0077P_____Name:__Aadit Deshpande_____

2. Mention the names of the Submitted files (Include Stage-1 and Stage-2 both)
   1) ast.c
   2) ast.h
   3) astDef.h
   4) driver.c
   5) lexer.c
   6) lexer.h
   7) ll1Grammar.c
   8) lookupTable.c
   9) parser.c
   10) parser.h
   11) parserDef.h
   12) predParseTable.c
   13) stack.c
   14) symbolTable.c
   15) symbolTable.h
   16) symbolTableDef.h
   17) typeChecker.c
   18) typeChecker.h
   19) grammar.txt, First.txt, Follow.txt
   20) TestCases: p1-p4.txt, s1-s5.txt, c1-c8.txt
   21) op.txt

3. Total number of submitted files: _____20 (Code Files)_____ (All files should be in **ONE** folder named exactly as Group number)
4. Have you mentioned names and IDs of all team members at the top of each file (and commented well)? (Yes/ no) ____YES_____ [Note: Files without names will not be evaluated]
5. Have you compressed the folder as specified in the submission guidelines? (yes/no)_____YES_____

6. **Status of Code development**: Mention 'Yes' if you have developed the code for the given module, else mention 'No'.
   a. Lexer (Yes/No): _____YES_____
   b. Parser (Yes/No):_____YES_____
   c. Abstract Syntax tree (Yes/No):_____YES_____
   d. Symbol Table (Yes/ No):_____YES_____
   e. Type checking Module (Yes/No):_____YES_____
   f. Semantic Analysis Module (Yes/ no):_____YES_____(reached LEVEL __3__ as per the details uploaded)
   g. Code Generator (Yes/No):_____NO_____

7. **Execution Status**:

a. Code generator produces code.asm (Yes/ No):_____No_____

b. code.asm produces correct output using NASM for testcases (C#.txt, #:1-11): _____No_____

c. Semantic Analyzer produces semantic errors appropriately (Yes/No):_____Yes_____

d. Static Type Checker reports type mismatch errors appropriately (Yes/ No):_____Yes_____

e. Dynamic type checking works for variant records with tagged union and reports errors on executing code.asm (yes/no):_____No_____

f. Symbol Table is constructed (yes/no)____Yes_____and printed appropriately (Yes /No):_____Yes_____

g. AST is constructed (yes/ no) _____Yes_____and printed (yes/no) _____Yes_____

h. Name the test cases out of 17 as uploaded on the course website for which you get the segmentation fault (p#.txt ; # 1-4, s$.txt; $ 1-5, and c@.txt ; @:1-8):_____None_____

8. **Data Structures** (Describe in maximum 2 lines and avoid giving C definition of it)

a. ***AST node structure***___Has the Fields: AST Node Type, Grammar Symbol, Lexeme, Line number and isUnion. Implemented as an N-ary Tree Node (Parent, First Child, Sibling)

_____

b. ***Symbol Table structure***:____3 sub tables for Identifiers, Record/Union and Function, with hash-chained nodes (3 types corresponding to each type of table)_____

c. ***Record type expression structure***:____ _____

d. ***Data structure for global variables***: _____Boolean global in "Identifier Node"_____

e. ***Variant record type expression structure:***_____ Boolean isVariant in the "Record/Union Node____

f. ***Input parameters type structure***:__ Token, AST Node Type, pointer to next parameter (linked list)__

g. ***Output parameters type structure***:_>same as input parameters>_____

h. Structure for maintaining the three address code(if created) :_____---_____

i. Any other interesting data structures used :_____

9. **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.]

a. ***Variable not Declared :***___Symbol Table Entry retrieved is empty_____

b. ***Multiple declarations***: ___Symbol Table Entry already for lexeme already exists_____

c. ***Number and type of input and output parameters***:___Checked in Pass 4 of AST traversal, while making Function Table___

d. **assignment of value to the output parameter in a function:** _Matching type and number of outputs of function call with symbol table. _____

e. function call semantics: Type checking and number matching the input and output parameters in function call with the function node in symbol table._____

f. ***static type checking*** :_____Referred to Symbol Table to compare AST Types_____

g. return semantics:_Checking if return parameters are assigned a vale and the type and lexeme is the same as the function output paarmeters._____

h. **Recursion :**_____No Recursion allowed in functions_____

i. module overloading:__implemented operator overloading_____

j. **if-then-else semantics :____**Implemented in AST with correct syntactic structure_____

k. **handling offsets for local variables** (starting with 0, integer size =2, real size =4 for symbol table purpose):__Each Function has local offset starting from 0, counting input/output parameters first then counting for local declarations of variables_____
_____

l. **handling offsets for formal parameters**:__iterate through parameter list and set offsets (starting from 0)_____

m. **handling global variable declaration over local variables and input-output parameters**:_overwrite in symbol table entry with global variable declaration_____
_____

n. Record semantics and static type checking: ____checking nested records attributes matching with the symbol table._____
_____

o. Variant record semantics and dynamic type checking: __(Not handled)_____
_____

p. **Scope of variables and their visibility** :_____stored and handled with identifier table and function table_____

q. handling nesting depth of variables in Boolean expression in while loop for assignment of an expression to one of the guard variables:_____checking if the comparison in a while loop is performed for variables that have been assigned a value in symbol table entry_____
_____

10. Compiler passes description (Mention the details of information collected/populated/worked upon at each traversal of the whole AST):

a. Pass 1: _collect names of records and unions_____
b. Pass 2: ____map alias of constructed datatype to actual name_____
c. Pass 3: ____creates record and union table traversing fields of all the records_____
d. Pass 4: ____width calculation for record and union recursively and storing the identifiers_____
e. Extra Pass:

11.* Code Generation: *(Not implemented)

a. NASM version as specified earlier used (Yes/no):_____---_____
b. Used 32-bit or 64-bit representation:_____---_____
c. For your implementation: 1 memory word = _____---_____(in bytes)
d. Mention the names of major registers used by your code generator:
   - For base address of an activation record: _---_____
   - for stack pointer:_____---_____
   - others (specify):_____---_____
e. Mention the physical sizes of the integer and real data as used in your code generation module
   size(integer): _____---_____(in words/ locations), _____---_____(in bytes)
   size(real): _____---_____(in words/ locations), _____---_____(in bytes)

f. How did you implement functions calls?(write 3-5 lines describing your model of implementation)
_____
_____(N.A.)_____
_____

g. Specify the following:
   - Caller's responsibilities:_____---_____
   - Callee's responsibilities:_____---_____

h. How did you maintain return addresses? (write 3-5 lines): _____
_____
_____(N.A.)_____
___
_____

i. How have you maintained parameter passing? How were the statically computed offsets of the parameters used by the callee? _____---_____

j. What have you included in the activation record size computation? (local variables, parameters, both):
_____---_____

k. Choice of registers (your manually selected heuristic only) _____
_____---_____

l. Which primitive data types have you handled in your code generation module?(Integer and real):
_____---_____

m. Where are you placing the temporaries in the activation record of a function? _____
_____---_____

n. Write your method of code generation for dynamic type checking for tagged union data type. _____
_____---_____

12. **Compilation Details**:
    a. Makefile works (yes/No):_____YES_____
    b. Code Compiles (Yes/ No):_____YES_____
    c. Mention the .c files that do not compile:_____None_____
    d. Any specific function that does not compile:_____None_____
    e. Ensured the compatibility of your code with the specified versions [GCC, UBUNTU, NASM] (yes/no)_____YES_____

13. Execution time for compiling the test cases [type checking (p1-p4.txt), semantic analyses including symbol table creation (s1-s5.txt), and code generation (c1-c8.txt)] :
    i. p1.txt (in ticks) _____ and (in seconds) _____
    ii. p2.txt (in ticks) _____ and (in seconds) _____
    iii. p3.txt (in ticks) _____ and (in seconds) _____
    iv. p4.txt (in ticks) _____ and (in seconds) _____
    v. s1.txt (in ticks) _____ and (in seconds) _____

vi. s2.txt (in ticks) _____ and (in seconds) _____

vii. s3.txt (in ticks) _____ and (in seconds) _____

viii. s4.txt (in ticks) _____ and (in seconds) _____

ix. s5.txt (in ticks) _____ and (in seconds) _____

x. c1.txt (in ticks) _____---_____ and (in seconds) _____---_____

xi. c2.txt (in ticks) _____---_____ and (in seconds) _____---_____

xii. c3.txt (in ticks) _____---_____ and (in seconds) _____---_____

xiii. c4.txt (in ticks) _____---_____ and (in seconds) _____---_____

xiv. c5.txt (in ticks) _____---_____ and (in seconds) _____---_____

xv. c6.txt (in ticks) _____---_____ and (in seconds) _____---_____

xvi. c7.txt (in ticks) _____---_____ and (in seconds) _____---_____

xvii. c8.txt (in ticks) _____---_____ and (in seconds) _____---_____

14. **Driver Details**: Does it take care of the **ELEVEN** options specified earlier?(yes/no):_____NO_____
(All 10, Except Code Generation)
15. Specify the language features your compiler is not able to handle (in maximum one line)
___NASM Code Generation, Variant Records Dynamic Checking_____
16. Are you availing the lifeline (Yes/No): _____NO_____
17. 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]
_____(Not Applicable)_____
_____
18. **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
19. **Any other point you wish to mention:** __We have implemented comprehensive functionality in the Symbol Table for handling nested records. Our Type Checking is very thorough and is capable of handling operations involving 2 records or a Record and a Scalar._____
20. Declaration: We, Preetika Verma, Pritika Ramu, Nandan Parikh, Sneha, and Aadit Deshpande declare that we have put our genuine efforts in creating the compiler project code and have submitted the code developed only by our group. 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. [Write your ID and names below]

ID__2019A7PS0088P _____ Name:__ Preetika Verma _____
ID__2019A7PS1140P _____ Name:__ Pritika Ramu _____
ID__2019A7PS0097P _____ Name:_ Nandan Parikh _____
ID__2019A7PS0042P _____ Name:__Sneha_____
ID__2019A7PS0077P _____ Name:__Aadit Deshpande_____

Date: ___16 April 2022_____

Should not exceed 6 pages.