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

Compiler Construction (CS F363)
II Semester 2022-23
Compiler Project (Stage-2 Submission)
Coding Details
(April 12, 2023)
Group number: 6

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: 2020A7PS0025P Name: Shaz Furniturewala
ID: 2020A7PS0067P Name: Niveditha Kovilakath
ID: 2020A7PS0111P Name: Anshika Gupta
ID: 2020A7PS0112P Name: Sarthak Agarwal
ID: 2020A7PS1684P Name: Shreyas Sesham

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

1. adt.c
2. adt.h
3. adtDef.h
4. ast.c
5. ast.h
6. astDef.h
7. astadt.c
8. astadt.h
9. astadtDef.h
10. codeGen.c
11. codeGen.h
12. codeGenDef.h
13. driver.c
14. grammarHash.c
15. grammarHash.h
16. grammarHashDef.h
17. keywordTable.c
18. keywordTable.h
19. keywordTableDef.h
20. lexer.c
21. lexer.h
22. lexerDef.h
23. ll1_gram.c
24. ll1_gram.h
25. ll1_gramDef.h
26. parser.c
27. parser.h
28. parserDef.h
29. symbolTable.c
30. symbolTable.h
31. symbolTableDef.h
32. typeChecking.c
33. grammar.txt
34. firstReadable.txt
35. followReadable.txt
36. MakeFile

1. Total number of submitted files: 35 (All files should be in **ONE** folder named exactly as Group number)
2. 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]
3. Have you compressed the folder as specified in the submission guidelines? (yes/no) **Yes**
1. **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 ____ as per the details uploaded)
g. Code Generator (Yes/No): **No**

1. **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 arrays 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 21 as uploaded on the course website for which you get the segmentation fault (t#.txt ; # 1-10 and c@.txt ; @:1-11): **1-10 None, Codegen not implemented**
2. **Data Structures** (Describe in maximum 2 lines and avoid giving C definition of it)
    a. AST node structure_____**contains pointer to corresponding parse tree node, parent, right sibling and leftmost child, along with the TERMINAL or NON-TERMINAL attribute, and the id of the node (based on the enum defined in stage 1)**_____
    b. **Symbol Table structure:** Is a struct containing table name, the latest offset, table width, nesting level, a pointer to the parent symbol table, an array of symbol records, and a pointer to a temporary table and two integers containing the scope. The corresponding symbol record contains the name isScope, a pointer to the symbol table called scopePointer, a symbol table entry type, a pointer to the parent table, a pointer to the next entry, entry datatype, width , offset, booleans, occupied, isfuncdef, isfuncdecl, funcCall, and two linkedlists for input_plist and output_plist
    c. **array type expression structure:** Contains a primitive datatype, a boolean for isDynamic, a lower bound and an upper bound in the form of a union between an integer and a variable.
    d. **Input parameters type structure:** A linkedlist of nodes where there leftChild of each node is the type and rightSibling is next parameter
    e. **Output parameters type structure**: A linkedlist of nodes where there leftChild of each node is the type and rightSibling is next parameter
    f. **Structure for maintaining the three address code(if created):** NA
1. **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 empty
    b. Multiple declarations: symbol table entry already found populated
    c. Number and type of input and output parameters: check between the function definition and function use
    d. assignment of value to the output parameter in a function:
    e. function call semantics: function declaration is above function call only when the function definition is later
    f. static type checking: type-mismatch
    g. return semantics:
    h. Recursion:
    i. module overloading:
    j. 'switch' semantics:
    k. 'for' and 'while' loop semantics:
    l. handling offsets for nested scopes:
    m. handling offsets for formal parameters:
    n. handling shadowing due to a local variable declaration over input parameters:
    o. array semantics and type checking of array type variables:
    a. Scope of variables and their visibility:
    b. computation of nesting depth:
2. Code Generation:

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):

a. Mention the physical sizes of the integer, real and boolean data as used in your code generation module

size(integer): _____(in words/ locations), _____(in bytes)
size(real): _____(in words/ locations), _____(in bytes)
size(booelan): _____(in words/ locations), _____(in bytes)

a. How did you implement functions calls?(write 3-5 lines describing your model of implementation):
a. Specify the following:
   - Caller's responsibilities:
   - Callee's responsibilities:

a. How did you maintain return addresses? (write 3-5 lines):
a. How have you maintained parameter passing? How were the statically computed offsets of the parameters used by the callee? _____
b. How is a dynamic array parameter receiving its ranges from the caller? _____
c. What have you included in the activation record size computation? (local variables, parameters, both):
_____
d. register allocation (your manually selected heuristic) :_____
_____
e. Which primitive data types have you handled in your code generation module?(Integer, real and boolean):_____
f. Where are you placing the temporaries in the activation record of a function? _____
_____

1. **Compilation Details**:
   a. Makefile works (yes/No):__Yes___
   b. Code Compiles (Yes/ No):___Yes_____
   c. Mention the .c files that do not compile:_____
   d. Any specific function that does not compile:_____
   e. Ensured the compatibility of your code with the specified versions [GCC, UBUNTU, NASM] (yes/no) Yes
2. Execution time for compiling the test cases [lexical, syntax and semantic analyses including symbol table creation, type checking and code generation] :
   i. t1.txt (in ticks) _____ and (in seconds) _0.042_____
   ii. t2.txt (in ticks) _____ and (in seconds) _0.040_____
   iii. t3.txt (in ticks) _____ and (in seconds) __0.067_____
   iv. t4.txt (in ticks) _____ and (in seconds) _0.058_____
   v. t5.txt (in ticks) _____ and (in seconds) __0.074_____
   vi. t6.txt (in ticks) _____ and (in seconds) __0.115_____
   vii. t7.txt (in ticks) _____ and (in seconds) ____0.100_____
   viii. viii. t8.txt (in ticks) _____ and (in seconds) ___0.116_____
   ix. t9.txt (in ticks) _____ and (in seconds) __0.152_____

x. t10.txt (in ticks) _____ and (in seconds) __0.055_____

1. **Driver Details**: Does it take care of the **TEN** options specified earlier?(yes/no):_____
2. Specify the language features your compiler is not able to handle (in maximum one line)

_____

1. Are you availing the lifeline (Yes/No): _No____
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]

_____
_____

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: We, ___Shaz Furniturewala, Niveditha Kovilath, Anshika Gupta, Sarthak Agarwal,Shreyas Shesam (your names) 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: 2020A7PS0025P	Name: Shaz Furniturewala
ID: 2020A7PS0067P	Name: Niveditha Kovilath
ID: 2020A7PS0111P	Name: Anshika Gupta
ID: 2020A7PS0112P	Name: Sarthak Agarwal
ID: 2020A7PS1648P	Name: Shreyas Shesam

Date: 12.04.2023 Group number: 6

Should not exceed 6 pages.