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

Batch No. :

**DEPARTMENT OF COMPUTER SCIENCE AND INFORMATION SYSTEMS**

**Compiler Construction (CS F363)**

Group Number

10

**II Semester 2021-22**

**Compiler Project (Stage-2 Submission)**

**Coding Details**

**(April 17, 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:\_\_\_\_2019A7PS0294P\_\_\_\_\_\_\_\_\_\_\_\_Name:\_\_\_\_\_Akash.S.Revankar\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ID:\_\_\_\_2019A7PS0070P\_\_\_\_\_\_\_\_\_\_\_\_Name:\_\_\_\_\_Hemant Singh Sisodiya\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ID:\_\_\_\_2019A7PS0022P\_\_\_\_\_\_\_\_\_\_\_\_Name:\_\_\_\_\_\_Harsh Butani\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ID:\_\_\_\_2019A7PS0033P\_\_\_\_\_\_\_\_\_\_\_\_\_Name:\_\_\_\_\_Siddharth Upadhyay\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ID:\_\_\_\_2019A7PS0100P\_\_\_\_\_\_\_\_\_\_\_\_\_Name:\_\_\_\_\_Mohit Sharma\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

1\_\_ast.c\_ 7\_\_grammar.txt\_ 13\_\_makefile\_ 19\_parser.c\_

2\_\_ast.h\_ 8\_\_hashtable.c\_ 14\_\_nonTerminals.txt\_ 20\_parser.h\_

3\_\_DFA.pdf\_ 9\_hashtable.h\_\_ 15\_\_p1.txt\_ 21\_parserDef.h\_

4\_\_driver.c\_\_ 10\_\_lexer.c\_ 16\_\_p2.txt\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 22\_s1.txt\_\_

5\_\_driver.h\_ 11\_\_lexer.h\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 17\_\_p3.txt\_ 23\_s2.txt\_

6\_\_FIRST\_AND\_FOLLOW\_SETS.pdf\_ 12\_lexerDef.h\_ 18\_p4.txt\_ 24\_s3.txt\_ 25. \_s4.txt\_

26 \_\_s5.txt\_\_ 27\_\_semantic\_analyzer.c\_\_ 28\_semantic\_analyzer.h 29\_SEMANTIC\_RULES\_FOR\_AST.pdf

30 \_set.c\_ 31 \_set.h\_ 32 \_stack.c\_ 33\_stack.h\_ 34\_testcase1.txt\_ 35\_testcase2.txt\_ 36\_testcase3.txt

37\_testcase4.txt 38\_testcase5.txt 39\_tokens.txt 40\_tree.c 41\_tree.h 42\_coding details stage 2.docx

43\_c1.txt\_ 44\_c2.txt\_ 45\_c3.txt\_ 46\_c4.txt\_ 47\_c5.txt\_ 48\_c6.txt\_ 49\_c7.txt\_ 50 c8.txt

1. Total number of submitted files: \_\_\_**50**\_\_\_\_\_\_ (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**\_\_\_\_\_\_\_\_\_\_\_\_
4. **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**\_\_\_\_\_\_\_
5. **Execution Status**:
   1. Code generator produces code.asm (Yes/ No):\_\_\_**NO**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
   2. code.asm produces correct output using NASM for testcases (C#.txt, #:1-11): \_\_\_\_**NO**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
   3. Semantic Analyzer produces semantic errors appropriately (Yes/No):\_\_**YES**\_\_\_\_\_\_\_\_\_\_
   4. Static Type Checker reports type mismatch errors appropriately (Yes/ No):\_\_**YES**\_\_\_\_\_\_\_\_\_\_\_\_
   5. Dynamic type checking works for variant records with tagged union and reports errors on executing code.asm (yes/no):\_\_\_\_\_\_\_\_\_\_\_\_**no**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
   6. Symbol Table is constructed (yes/no)\_\_\_\_**yes**\_\_\_\_\_and printed appropriately (Yes /No):\_\_\_\_\_\_\_\_\_**yes**\_\_\_\_\_\_\_
   7. AST is constructed (yes/ no) \_\_\_**yes**\_\_\_\_\_\_\_\_\_and printed (yes/no) \_\_\_**yes**\_\_\_\_\_\_\_
   8. 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**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
6. **Data Structures** (Describe in maximum 2 lines and avoid giving C definition of it)
   1. AST node structure\_\_\_\_**AST is an n-ary tree. Node structure: pointers to parent, leftmostchild, rightmostchild and brother. Number of children, symbol token. Extra Args(for while loop), visited, scope symbol table pointer, pointer to type structure of enclosing function.**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

* 1. Symbol Table structure:\_**members include: a hashtable, enclosing function name, current offset, pointers to parent, leftmost, rightmost and brother symbol tables.**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

* 1. Record type expression structure:\_**we have a structure called type which has common members like name, width, offset, is\_assigned and a union of 4 structs namely Record, Union, Alias and Function. The members of Record structure include: ^isTaggedUnion, pointers to type structure of ^tag and ^variantField, a linked list of aliases, a linked list of its members, record name and the currentoffset. (^ these fileds are used only when the record is of type variant record)**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
  2. Data structure for global variables: \_\_**the type structure explained above is used for global variables as well. We insert global variable entries in the root symbol table**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
  3. Variant record type expression structure:\_\_**the structure of variant record is same as that of record type. The bool isTaggedUnion is used to differentiate between the two. The type pointers tag and variantField are used to access the types of the members**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
  4. Input parameters type structure:\_\_**Input parameters are stored as a linked list within the structure for the Functions. The structure for Function includes linked list for input params, linked list for output params, function name, current offset and offset of params.**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
  5. Output parameters type structure:\_**as explained above**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
  6. Structure for maintaining the three address code(if created) :\_\_\_**N.A.**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
  7. Any other interesting data structures used :\_\_\_**Union: it includes linked list of aliases, linked list of members, name of union and current offset. Alias: it includes two members: it’s own name and the name of the constructed type of which it is an alias**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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.]
   1. Variable not Declared :\_**first\_we\_check\_for\_variables\_in\_the\_root\_symbol\_table\_,if\_not\_found\_check\_in\_the\_local\_variablesof\_function\_than\_check\_for\_the\_input\_and output\_parameters\_of\_the\_function\_if\_not\_found\_anywhere\_report\_the\_error**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
   2. Multiple declarations: \_**Before\_inserting\_the\_variables\_we\_first\_check\_for\_variables\_in\_the\_root\_symbol\_table\_,if\_not\_found\_check\_in\_the\_local\_variables of\_function\_than\_check\_for\_the\_input\_and output\_parameters\_of\_the\_function\_if\_found\_anywhere\_report\_the\_error**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
   3. Number and type of input and output parameters:\_**we\_traverse\_the\_list\_of\_parameters\_of\_the\_input\_and\_output\_parametes\_declared\_during\_function\_\_definition\_as\_well\_as\_the\_list\_of\_the\_functioncall,\_one\_by\_one\_we\_check\_\_the\_parametes\_if\_the\_type\_does\_not\_match\_print the type error if either of the actual parameter list end or the functioncall parameter ends first than print the number of parameters does not match.**
   4. assignment of value to the output parameter in a function : **this is handled during the assignment verification, if we assign value to any of the variable than, we check if it is the part of output parameter list,if yes than mark it as assigned.**
   5. function call semantics:\_\_**we check if the count and types of the formal and actual match. If there’s a mismatch in either, return error**
   6. static type checking :\_\_**primitive types: check if the name parameter of type structure matches, record and union: check if the name of the constructed data types match or if one is the alias of the other. If both are of alias type, recursively get the base type using the aliasOf parameter and check if the base types match.**
   7. return semantics:\_\_\_ **we check if the count and types of the output parameters and those returned by the function match. If there’s a mismatch in either, return error**
   8. Recursion :\_\_\_**Whenever the function is called we check for the function name and the actual function in which it is called, if the name matches then recursion is present** \_\_\_\_
   9. module overloading:\_\_**whenever the new function is defined then check if the function with the same name already exist or not, if the function with same name exists then it is a case of overloading, else insert the function definition in the root symbol table**
   10. if-then-else semantics :\_\_**we use divide and conquer to recursively check if the Boolean expression semantics match.**
   11. handling offsets for local variables (starting with 0, integer size =2, real size =4 for symbol table purpose):\_\_**to handle the offset we maintain the offset variable for each scope which on any declaration increases it value by the width of the variable declared.** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
   12. handling offsets for formal parameters**:\_\_ the formal parameters have the offset starting from the 0, first the offset for the input parameters is set then for the output parameters then for the local variables**
   13. handling global variable declaration over local variables and input-output parameters:\_\_\_**whenever the global variable is declared we maintain its entry in the root symbol table. If a local variable or function parameter is declared with the same name, we report error.**
   14. Record semantics and static type checking: \_**we recursively check if the members are a part of the record whenever used. If not, report error. Same method is used for getting their types. For type checking, the names of the record or aliases are compared. In case both are aliases we get the base type and compare their names**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

* 1. Variant record semantics and dynamic type checking: \_\_**variant record is identified using the isTaggedUnion filed. Dynamic type checking hasn’t been implemented**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

* 1. Scope of variables and their visibility :\_\_\_**if a variable is declared locally than its entry will be present in the current function/scope symbol table, so it can not be used anywhere outside, but for the global variable the entry is present in the root symbol table from where it is visible to every scope/function**
  2. handling nesting depth of variables in Boolean expression in while loop for assignment of an expression to one of the guard variables:\_**if a variable comes on the lhs of an assignment or is part of a read call, we set the isAssigned member of its type structure which is then used while verifying while loop semantics.**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. **Compiler passes description (Mention the details of information collected/populated/worked upon at each traversal of the whole AST):**
   1. Pass 1: \_\_**generate a map, ruidToNode which maps from the record/union or alias name to it’s corresponding definition node in the ast. This is done to handle global visibility of constructed data type definitions**
   2. Pass 2: **populate the symbol table**
   3. Pass 3: \_\_**Verify various construct semantics**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
   4. Pass 4: \_\_**Not Applicable**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
2. **Code Generation:** 
   1. NASM version as specified earlier used (Yes/no):\_\_\_\_\_**no**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
   2. Used 32-bit or 64-bit representation:\_\_\_\_\_\_\_\_\_\_\_\_**n.a.**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
   3. For your implementation: 1 memory word = \_\_\_\_\_\_**n.a**\_\_\_\_\_\_\_\_\_\_\_\_\_\_(in bytes)
   4. Mention the names of major registers used by your code generator:

* For base address of an activation record: \_\_\_\_**n.a.**\_\_\_\_\_\_\_\_\_\_\_\_\_\_
* for stack pointer:\_\_\_\_\_\_\_\_\_**n.a.**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
* others (specify):\_\_\_\_\_\_\_\_\_\_**n.a.**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
  1. Mention the physical sizes of the integer and real data as used in your code generation module

size(integer): \_\_\_\_\_\_\_\_\_\_**n.a.**\_\_\_\_\_\_\_\_\_\_\_\_\_(in words/ locations), \_\_\_\_\_\_**n.a.**\_\_\_\_\_\_\_\_\_(in bytes)

size(real): \_\_\_\_\_\_\_\_\_**n.a.**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(in words/ locations), \_\_\_\_\_\_**n.a.**\_\_\_\_\_\_\_\_\_(in bytes)

* 1. How did you implement functions calls?(write 3-5 lines describing your model of implementation) \_\_\_\_\_\_\_\_\_\_\_\_\_\_**n.a.**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

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

* 1. Specify the following:
     + Caller's responsibilities:\_\_\_\_\_\_\_\_\_\_**n.a.**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
     + Callee's responsibilities:\_\_\_\_\_\_\_\_\_**n.a.**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
  2. How did you maintain return addresses? (write 3-5 lines): \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**n.a.**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

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

* 1. How have you maintained parameter passing? How were the statically computed offsets of the parameters used by the callee? \_\_\_**n.a.**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
  2. What have you included in the activation record size computation? (local variables, parameters, both): \_\_\_\_\_\_\_\_\_**n.a.**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
  3. Choice of registers (your manually selected heuristic only) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**n.a.**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
  4. Which primitive data types have you handled in your code generation module?(Integer and real): \_\_\_\_\_\_\_\_\_\_\_\_\_\_**n.a.**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
  5. Where are you placing the temporaries in the activation record of a function? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_**n.a.**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
  6. Write your method of code generation for dynamic type checking for tagged union data type. \_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**n.a.**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. **Compilation Details**:
   1. Makefile works (yes/No):\_\_**yes**\_\_\_\_\_\_\_\_\_
   2. Code Compiles (Yes/ No):\_\_\_\_\_**yes**\_\_\_\_\_\_\_\_\_
   3. Mention the .c files that do not compile:\_\_\_\_\_\_\_\_\_\_\_**none**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
   4. Any specific function that does not compile:\_\_\_\_\_\_\_\_**none**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
   5. Ensured the compatibility of your code with the specified versions [GCC, UBUNTU, NASM] (yes/no)\_\_\_\_\_**yes**\_\_\_\_\_\_\_
2. 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)] :
   * 1. p1.txt (in ticks) \_\_\_\_\_\_2626\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ and (in seconds) \_\_0.002626\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
     2. p2.txt (in ticks) \_\_\_\_3445\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ and (in seconds) \_\_0.003445\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
     3. p3.txt (in ticks) \_\_\_\_\_2594\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ and (in seconds) \_\_\_\_0.002594\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
     4. p4.txt (in ticks) \_\_\_\_\_\_3715\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ and (in seconds) \_\_\_0.003715\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
     5. s1.txt (in ticks) \_\_\_\_\_\_2648\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ and (in seconds) \_\_\_\_0.002648\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
     6. s2.txt (in ticks) \_\_\_\_\_\_3795\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ and (in seconds) \_\_\_0.003795\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
     7. s3.txt (in ticks) \_\_\_\_\_\_2247\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ and (in seconds) \_\_0.002247\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
     8. s4.txt (in ticks) \_\_\_\_\_\_4682\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ and (in seconds) \_\_0.004682\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
     9. s5.txt (in ticks) \_\_\_\_\_1624\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ and (in seconds) \_\_\_0.001624\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
     10. c1.txt (in ticks) \_\_\_\_\_\_\_**n.a**\_\_\_\_\_\_\_\_\_\_\_\_\_\_ and (in seconds) \_\_\_\_**n.a**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
     11. c2.txt (in ticks) \_\_\_\_\_\_\_\_\_\_ **n.a** \_\_\_\_\_\_\_\_\_\_\_ and (in seconds) \_\_\_ **n.a** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
     12. c3.txt (in ticks) \_\_\_\_ **n.a** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ and (in seconds) \_\_\_ **n.a** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
     13. c4.txt (in ticks) \_\_\_\_\_\_\_ **n.a** \_\_\_\_\_\_\_\_\_\_\_\_\_\_ and (in seconds) \_\_\_\_ **n.a** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
     14. c5.txt (in ticks) \_\_\_\_\_\_\_\_\_ **n.a** \_\_\_\_\_\_\_\_\_\_\_\_ and (in seconds) \_\_\_\_\_\_\_ **n.a** \_\_\_\_\_\_\_\_\_\_\_\_\_
     15. c6.txt (in ticks) \_\_\_\_\_ **n.a** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ and (in seconds) \_\_\_ **n.a** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
     16. c7.txt (in ticks) \_\_\_\_\_\_\_\_ **n.a** \_\_\_\_\_\_\_\_\_\_\_\_\_ and (in seconds) \_\_\_\_\_ **n.a** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
     17. c8.txt (in ticks) \_\_\_\_\_ **n.a** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ and (in seconds) \_\_\_\_\_\_\_ **n.a** \_\_\_\_\_\_\_\_\_\_\_\_\_
3. **Driver Details**: Does it take care of the **ELEVEN** options specified earlier?(yes/no):\_**YES**\_\_\_\_\_\_\_\_\_\_
4. Specify the language features your compiler is not able to handle (in maximum one line)

\_**All features excluding the code generation phase have been implemented and are working correctly**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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]

\_\_\_\_\_\_\_\_\_**not implemented**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

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: \_\_**In large testcases, printing the AST (in pre-order) might lead to stack smashing depending on the machine being used. The parse tree is being printed in in-order fashion to avoid the same.**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

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

1. Declaration:We,\_\_**Hemant Singh Sisodiya, Akash.S.Revankar, Harsh Butani, Mohit Sharma, Siddharth Upadhyay**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (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\_\_\_\_\_\_2019A7PS0294P \_\_\_\_\_\_\_\_\_ Name:\_\_\_\_\_\_\_\_ Akash.S.Revankar \_\_\_\_\_\_\_\_

ID\_\_\_\_\_\_2019A7PS0070P \_\_\_\_\_\_\_\_\_\_ Name:\_\_\_\_\_\_\_ Hemant Singh Sisodiya\_\_\_\_\_

ID\_\_\_\_\_\_2019A7PS0022P \_\_\_\_\_\_\_\_\_\_ Name:\_\_\_\_\_\_\_ Harsh Butani \_\_\_\_\_\_\_\_\_\_\_\_\_

ID\_\_\_\_\_\_2019A7PS0033P \_\_\_\_\_\_\_\_\_\_ Name:\_\_\_\_\_\_ Siddharth Upadhyay \_\_\_\_\_\_\_\_

ID\_\_\_\_\_\_2019A7PS0100P \_\_\_\_\_\_\_\_\_\_ Name:\_\_\_\_\_\_\_ Mohit Sharma \_\_\_\_\_\_\_\_\_\_\_

Date: \_\_17/04/2022\_\_\_\_\_\_\_\_\_\_\_\_

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

Should not exceed 6 pages.