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

Group No. 15

Compiler Construction (CS F363)
II Semester 2022-23
Compiler Project (Stage-1 Submission)
Coding Details
(March 2, 2023)

---

1. IDs and Names of team members

2.

ID: 2020A7PS0981P	Name: Utkarsh Darolia
ID: 2020A7PS0084P	Name: Tanveer Singh
ID: 2020A7PS0124P	Name: Shivam Abhay Pande
ID: 2020A7PS0980P	Name: Mithil Shah
ID: 2020A7PS0120P	Name: Kshitij Garg

3. Mention the names of the Submitted files :

1. driver.c	7. htDef.h	13. makefile
2. grammar.txt	8. lexer.c	14. parser.c
3. grammerSymbol.c	9. lexer.h	15. parser.h
4. grammerSymbol.h	10. lexerDef.h	16. parserDef.h
5. ht.c	11. linkedList.c	17. tree.c
6. ht.h	12. linkedList.h	18. tree.h

19. coding_details_stage_1.pdf
20. t1.txt-t6.txt (Provided by IC) 21. testcase1.txt-testcase6.txt (Testcases from language specification doc)

4. Total number of submitted files: **19 + 6 + 6 = 31** (All files should be in **ONE folder** named exactly as Group_#, # is your group number)

5. Have you mentioned your names and IDs at the top of each file (and commented well)? (Yes/ no) **Yes** [Note: Files without names will not be evaluated]

6. Have you compressed the folder as specified in the submission guidelines? (yes/no) **Yes**

7. Lexer Details:

[A]. Technique used for pattern matching: **Switch Case simulated DFA**.

[B]. DFA implementation (State transition using switch case, graph, transition table, any other (specify):

Combination of Switch Case and If-Else statements.

[C]. Keyword Handling Technique: **Lookup table implemented using Hash Table.**

[D]. Hash function description, if used for keyword handling: **The hash key is generated from the string input as key using 'Fowler-Noll-Vo 1-a' hash function and index is the modulus of hash key with respect to hashtable size.**

[E]. Have you used twin buffer? (yes/ no): **Yes.**

[F]. Lexical error handling and reporting (yes/No): **Yes.**

[G].Describe the lexical errors handled by you: **Unidentified Symbols, Identifier too long, Incorrect token syntax.**

[H].Data Structure Description for tokenInfo (in maximum two lines):

Structure with the fields token name, token lexeme, line number and token value. Token name is an enum of possible tokens, and token value is a union of int, float and bool.

[I]. Interface with parser: **getNextToken() function calls by the parser to receive the next token.**

8. Parser Details:

[A]. High Level Data Structure Description (in maximum three lines each, avoid giving C definitions used):

i. grammar : **an array of linked lists, each linked list defines a rule, linked list head is the non terminal at the LHS of rule, the next of each node determines the next grammar symbol on RHS of the rule.**

ii. parse table: **a 2d array of linked list nodes, where each node represents linked list head to the corresponding rule.**

iii. parse tree: (Describe the node structure also) **n-ary tree, node contains pointer to the first child, last child, parent, next sibling, previous sibling node, it also stores the token information and grammar symbol information**

iv. Parsing Stack node structure : **Stack nodes are linked list nodes that contain grammar symbols**

v. Any other (specify and describe)

Grammar Symbol: A struct which contains information about whether the symbol is terminal, nonterminal or epsilon, it also contains corresponding first and follow set of the non terminal

[B].Parse tree

i. Constructed (yes/no): **Yes**

ii. Printing as per the given format (yes/no): **Yes**

iii. Describe the order you have adopted for printing the parse tree nodes (in maximum two lines)

In order traversal, the first child is printed first, followed by the node itself, and then the rest of its children sequentially.

[C].Grammar and Computation of First and Follow Sets

i. Data structure for original grammar rules: **Linked List, head representing non terminal on LHS of the rule, and linked list nodes determine the grammar symbols on RHS of the rule.**

ii. FIRST and FOLLOW sets computation automated (yes /no): **Yes**

iii. Data structure for representing sets: **Integers, used as bitsets (Using bit manipulation, we are able to store both the first and follow set with just 2 integers (128 bits) for each rule)**

iv. Time complexity of computing FIRST sets: **O(Size_of_Grammer), where Size_of_Grammer = Number of Rules*Rule Length**

v. Name the functions (if automated) for computation of First and Follow sets: **computeFirstAndFollowSets()**

vi. If computed First and Follow sets manually and represented in file/function (name that) **NA**

[D]. Error Handling

i. Attempted (yes/ no): **Yes**

ii. Printing errors (All errors/ one at a time) : **All errors**

iii. Describe the types of errors handled

Lexical Errors, top of stack is terminal and not matching with the current token, corresponding grammar rule for non-terminal is not present in the parse table.

iv. Synchronizing tokens for error recovery (describe): **Synchronizing Sets (consisting of Follow Sets). Along with few special tokens such as BC, SEMICOL, END etc are used. To limit errors to a particular line or module we give errors until BC, SEMICOL, END and stop propagation of errors after that by using effectinve error handling heuristics.**

v. Total number of errors detected in the given testcase t6(with_syntax_errors).txt: **All 12 errors detected. (Propagation of errors is limited to the same line)**

9. 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 gcc version(yes/no): **Yes**

10. Driver Details: Does it take care of the options specified earlier(yes/no): **Yes**

11. Execution

[A]. status (describe in maximum 2 lines): **Everything works perfectly.**

[B]. Execution time taken for

- **t1.txt (in ticks) 98 and (in seconds) 0.000098**
- **t2.txt (in ticks) 158 and (in seconds) 0.000158**
- **t3.txt (in ticks) 363 and (in seconds) 0.000363**
- **t4.txt (in ticks) 699 and (in seconds) 0.000699**
- **t5.txt (in ticks) 947 and (in seconds) 0.000947**
- **t6.txt (in ticks) 828 and (in seconds) 0.000828**

[C]. Gives a segmentation fault with any of the test cases (1-6) uploaded on the course page. If yes, specify the testcase file name: **No**

12. Specify the language features your lexer or parser is not able to handle (in maximum one line): **Expression can contain a mixture of semantically incorrect arithmetic and logical expressions, which will be handled in semantics.**

13. Are you availing the lifeline (Yes/No): **No**

14. Declaration: We, Tanveer Singh, Utkarsh Darolia, Shivam Pande, Kshitij Garg and Mithil Shah, declare that we have put our genuine efforts into 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 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 name below]

ID: 2020A7PS0084P	Name: Tanveer Singh
ID: 2020A7PS0981P	Name: Utkarsh Darolia
ID: 2020A7PS0124P	Name: Shivam Abhay Pande
ID: 2020A7PS0120P	Name: Kshitij Garg
ID: 2020A7PS0980P	Name: Mithil Shah

Date: March 02, 2023

Should not exceed 4 pages.