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

Compiler Construction (CS F363)
II Semester 2023-24
Compiler Project
Coding Details
(March 5, 2022)

Group Number

30

---

1. **Team Members Names and IDs**
   ID 2021A7PS0939P____________________________Name: Anish Taori
   ID 2021A7PS2681P____________________________Name: Aditya Deshpande
   ID 2021A7PS0426P____________________________Name: Aaditya Kulkarni
   ID 2021A7PS2417P____________________________Name: Yash Kumar Kandoi
   ID 2021A7PS2538P____________________________Name: Roshan Bagla
   ID 2021A7PS2226P____________________________Name: Garvit Singhal
2. **Mention the names of the Submitted files:**

1___grammar.txt________	7________makefile_____	13_______testcase5.txt_________
2___lexer.h____________	8________driver.c______	14_______testcase6.txt_________
3___lexer.c_________	9________testcase1.txt___________	15_______testcase_700.txt_________
4___parser.h_________	10_______testcase2.txt__________	16_______lexerDef.h________
5___parserDef.h________	11_______testcase3.txt__________	17_______coding_details.pdf____
6___parser.c____________	12_______testcase4.txt___________	18_________________

3. **Total number of submitted files (including copy of the pdf file of this coding details pro format)** : ____17____
   (All files should be in ONE folder named Group_#)

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

5. **Lexer Details:**
   [A]. The technique used for pattern matching: __Hashing__
   [B]. Keyword Handling Technique: _Whenever we take a lexeme (TK_ID or TK_FUNID) as an input, we first check if the lexeme is already present in the predefined set of keywords in the symbol table. This way, we are prioritizing predefined keywords over the new lexemes._
   [C]. Hash function description, if used for keyword handling:__Not used__
   [D]. Have you used twin buffer? (yes/ no) ___YES____
   [E]. Error handling and reporting (yes/No):____YES___
   [F]. Describe the errors handled by you: __Handled 1. lexemes with unknown symbols
   2. lexemes with unknown patterns
   3. lexemes with token TK_ID and exceeding a length of 20 characters
   4. lexemes with token TK_FUNID and exceeding a length 30 characters.
   __
   [G]. Data Structure Description for tokenInfo (in maximum two lines): __`tokenInfo` is a structure consisting of pointers to character arrays (`char*`) representing the token (`token`) and its corresponding lexeme (`lexeme`), along with an integer (`int`) for the line number (`line_no`).___

6. **Parser Details:**

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

i. **grammar**: Our grammar is encoded as a two-dimensional integer array which is initialized using mapping non-terminals and terminals to integers (0-110). Each rule is represented as an array starting with a non-terminal, followed by terminals and non-terminals, and ending with '-1'. This structure populates our grammar matrix with various rule definitions.

ii. **FIRST and FOLLOW sets**: We have defined a structure 'FirstAndFollow' which holds 2 two-dimensional arrays `first` and `follow` representing the first and follow sets respectively for each grammar term. Each set contains the mapped integers of the terminals which belong to the set. Each set ends with -1 to mark its ending.

iii. **parse table:** For the parse table again we defined a 2D int array in which rows include the non terminals and columns include non terminals including dollar symbol. If a rule exists for a combination of the top of the stack term and the next input then we insert the index of that rule as the value of that particular array element.

iv. **parse tree: (Describe the node structure also)** Parse Tree is made up of parse nodes which has a structure which includes lexeme, mapped int value of the token, number of children of the node, lineNo, array of parse nodes which defines the children and parent parse node. The parse tree is generated in such a way that if there is a rule <program> ===> <otherfunctions> <mainfunctions> then <program>, <otherfunctions> and <mainfunctions> will be the nodes and <otherfunctions> and <mainfunctions> will be the children of <program> node.

v. **Any other (specify and describe)** _We have specified a static array of strings grammarTerms which includes all the terminals and non terminals and index at which a given terminal or non terminal is stored is the value with which it is defined in all other functions. Also we are maintaining a stack for parsing purposes.

[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)**
Inorder traversal has been adopted for printing the parse tree nodes in which the leftmost child gets printed first then the current node and then the rest of the siblings of the leftmost child.

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

i. **Data structure for original grammar rules** 2D int array
ii. **FIRST and FOLLOW sets computation automated (yes /no)**___YES___
iii. **Name the functions (if automated) for computation of First and Follow sets**__First and Follow are the function names for computation of First and Follow sets respectively.
iv. If computed First and Follow sets manually and represented in file/function (name that) __NA____

[D]. Error Handling

v. **Attempted (yes/ no):**____YES_____
vi. **Describe the types of errors handled** Handled: 1. Top of the stack does not match with incoming input token
2. There is no rule in the parser table for the particular combination of non terminal present at the top of stack and the incoming input token. In that case skip the token.
3. There exists syn in the parser table for the particular combination of non terminal present at the top of stack and the incoming input token. In that case the top of the stack is pop.

7. 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__

8. **Driver Details: Does it take care of the options specified earlier(yes/no)**:______YES_____
9. **Execution**
   [A]. **status (describe in maximum 2 lines):** The code executes without any segmentation fault on the given 7 test cases but in the testcase6.txt the number of syntax errors generated by our code is slightly different as compared to the given list of errors.
   [B]. **Gives segmentation fault with any of the test cases (1-6) uploaded on the course page. If yes, specify the testcase file name:** NO

10. **Specify the language features your lexer or parser is not able to handle (in maximum one line)**_According to us our lexer and parser are handling all the language features.

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

12. **Declaration**: We, Roshan Bagla, Garvit Singhal, Aaditya Kulkarni, Anish Taori, Aditya Deshpande and Yash Kandoi declare that we have put our genuine efforts in creating the compiler project code and have submitted the code developed only by us. 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 all of us in our team and we will accept the penalty as decided by the department of Computer Science and Information Systems, BITS, Pilani.

Your names and IDs

Name:ANISH TAORI	**ID: 2021A7PS0939P**
Name:GARVIT SINGHAL	**ID: 2021A7PS2226P**
Name:ROSHAN BAGLA	**ID: 2021A7PS2538P**
Name:ADITYA DESHPANDE	**ID: 2021A7PS2681P**
Name:AADITYA KULKARNI	**ID: 2021A7PS0426P**
Name:YASH KANDOI	**ID: 2021A7PS2417P**

Date: 05-03-2024

Not to exceed 3 pages.