CoPro:
Lexical Analyzer

T. Rohan - CS20BTECH11064
I. Rajasekhar - CS20BTECH11020
P. Vikas - CS20BTECH11037
S. Satvik - CS20BTECH11048
B. Revanth - CS20BTECH11007
J. Vamsi Preetham - CS20BTECH11058
G. Yagnavalkya - CS20BTECH11019

Lexical Analysis

- **First phase of a compiler.**
- **Input.cop → Tokens**
  - The lexer receives the cop file as an input
  - Removes whitespaces and comments
  - Converts the file into a series of tokens.
  - Returns all the valid tokens and also ignores the invalid ones.
  - These tokens vary from constants, identifiers to operators.

Lexer

Flex

- Our lexer has been written using flex
- ‘.cop’ files would be taken as the input for the lexer, which then tokenizes the file.
- The tokens would then be passed on to the parser as input (to be implemented in the later stages).

Implementation

- The input file would be given in the ‘.cop’ format.
- In the ‘lexer.l’ file, definitions and rules are defined for identifying each particular sequences of characters.
- Once the respective sequence is identified according to the precedence given, a token is assigned to that sequence.
- There also exist C functions within ‘lexer.l’ to respond to each sequence while assigning tokens.

Implementation

- When we type ‘make’ in the terminal, first a ‘lex.yy.c’ file would be created which contains the corresponding generated C code for the ‘lexer.l’ file.
- The ‘lex.yy.c’ file, takes in input from a ‘.cop’ file.
- It then goes through the file, iteratively assigning tokens to each sequence of characters
- It then prints the final tokens generated along with their token numbers into another file.

Test cases and Outputs

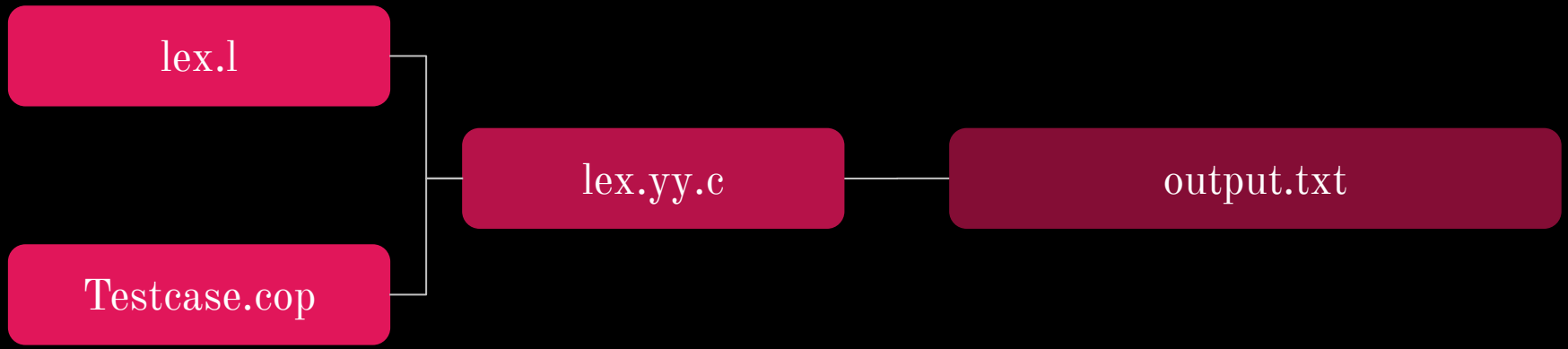
Test cases

- Inside the ‘Test case’ folder are five example codes of CoPro language.
- When we type ‘make’, after creating ‘lex.yy.c’, it goes through each of the test codes inside the folder, and tokenizes them.

Output

- The tokens would then be written onto the corresponding output numbered file, present in the ‘Outputs’ folder.

File Flow



Example- Sum of 2 Integers

.cop file-

```
main->int
‘
        int left = 21
        int right = 11
        int ans = left + right
        output: “sum of left and right is:”, ans
        exit : 0
’
```

Lexer output-

1 MAIN_FUN	10 VARIABLE	19 OUTPUT
2 PTR_OP	11 =	20 :
3 INT	12 D	21 STRING_LITERAL
4 SIN_QUO	13 INT	22 ,
5 INT	14 VARIABLE	23 VARIABLE
6 VARIABLE	15 =	24 EXIT
7 =	16 VARIABLE	25 :
8 D	17 ADD_OP	26 D
9 INT	18 VARIABLE	

Thank You