# Lexical Analyzer Report

N Sai Vamsi
CS21BTECH11038

## Section 1: Compilation steps

- First we go to the directory TP1 which contains the source file names as “source.l”
- Then we use the lex tool for generating lex.yy.c
- The command to do so is “lex source.l”
- Now we would have a file names lex.yy.c in our directory.
- To input the given code of program P which is a variant of C, first we should bring the input files to the same directory as source code
- Then we run the command gcc lex.yy.c which generates our Lexical analyser which would be named as “a.out”
- Running the following command “./a.out <name_of_input_file>” (EX:1.txt) will output 2 files.
- The file named “token_stream” contains the tokens generated by this lexical analyzer.
- The file named “C_code” contains the corresponding C-variant of our input program generated by this lexical analyser

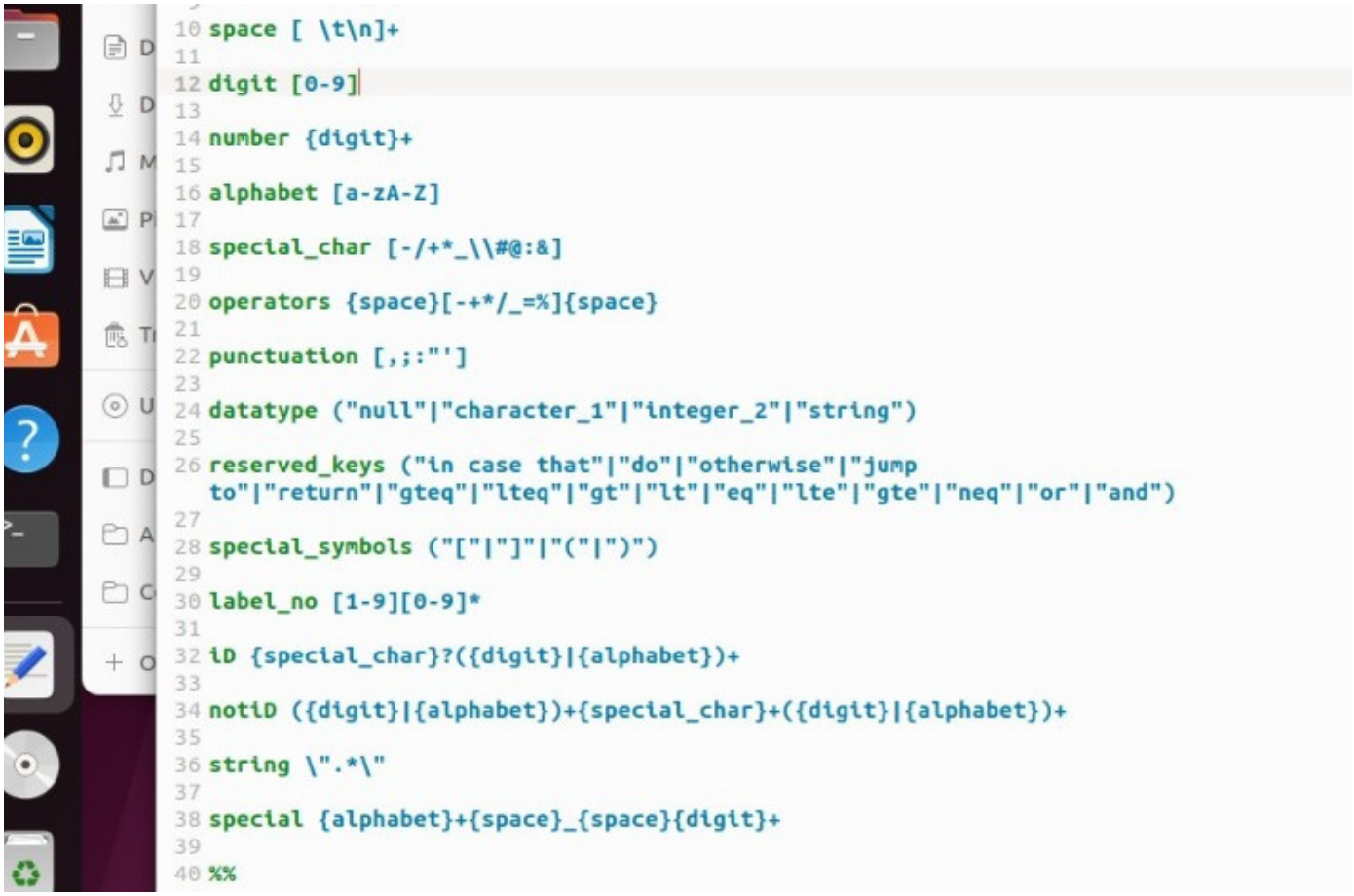
Following is the instructions set that I did on my linux terminal

```
$lex source.l
```

```
$gcc lex.yy.c
```

```
$./a.out 1.txt
```

## Regular expressions in my source code:



```
space [ \t\n]+

digit [0-9]

number {digit}+

alphabet [a-zA-Z]

special_char [-/+*_\\#@:&]

operators {space}[-+*/_=%]{space}

punctuation [,;:"']

datatype ("null"|"character_1"|"integer_2"|"string")

reserved_keys ("in case that"|"do"|"otherwise"|"jump
to"|"return"|"gteq"|"lteq"|"gt"|"lt"|"eq"|"lte"|"gte"|"neq"|"or"|"and")

special_symbols ("["|"]"|"("|")")

label_no [1-9][0-9]*

iD {special_char}?({digit}|{alphabet})+

notiD ({digit}|{alphabet})+{special_char}+({digit}|{alphabet})+

string \".*\"

special {alphabet}+{space}_{space}{digit}+

%%
```

The following are the regular expressions in my source code

**1. Space:**
This part makes up to the pattern that contains spaces, newlines and tabs in our source file

**2. Digit:**
This part makes up to the pattern that contains all the digits 0 to 9

**3. Number:**
This part makes up to the pattern that contains all the whole numbers including digits

**4. Alphabet:**
This part makes up to the pattern that contains capital and small alphabets and capital alphabets in English language, that is from a to z and A to Z(Note I made mistake on my 1st try by writing[a-zA-z] instead of [a-zA-Z] which would include some other charecters to our alphabet).

**5. Special_char:**
This part makes up to pattern that contains all the special characters(recognises only 1 special character). In this we wrote – in 1st place because the LEX tool may mistake it for hyphen symbol which has other meaning inside square brackets.

**6. Operators:**
This part makes up to the pattern of operators that are generally used in mathematics. Here once again – is written in the 1st . Now for the operator _ which makes up to nth root when followed by a integer constant n is written in another regular expression (special) to makes things little easier

**7. Punctuations:**
This part makes up to the pattern containing the punctuation , : and ;
I also included ‘ and “ but later I defined string and character literals so no need to include them

**8. Data type:**
This part makes up to the data types null,integer_2,character_1 and string

**9. Reserved keywords:**
This part makes up to the key words which csnnot be IDs as they have their own functionalities. These should be kept 1st in rules.

**10. Special symbols:**
Makes up to the patterns containing ( , ) , [ , ]

**11. Label number:**
It is similar to number but as pp1 is followed by only +ve integers we need to redefine them

**12. ID:**
This makes up to the pattern IDs that are used as variables and function names

**13. notID:**
this makes up to the pattern that contains special charecaters in the middle as these re not IDs, my LA throws error here and exits

**14. string:**
This makes up to the pattern of strings, that is all that is in between “ and “ and including them also

**15. Special*:**
This makes up to the pattern containg the following format x _ 2. Which is the nth root operator
And it outputs as pow(a,b) function ;

16. Finally, makes up to all the other remaining patterns and throws an error if we find them