Name: Fahad Ali, Registration ID: L1F22BSCS0421

Course: CC, Section: G5

Lexical Analyzer Project Report: Project Phase 1

Jugr Language

1. **Introduction and Language Design:**
   I designed and implemented a lexical analyzer scanner for a unique Mini C++ like language named **Jugr**. This scanner was built using **Flex** (Fast Lexical Analyzer Generator) and is responsible for tokenizing the source code, tracking line numbers, and reporting errors.
2. **Project Uniqueness:**
   I used by own unique keywords to make my language. Jugr is inspired from urdu word jugaar. As we sometimes do jugaar in programming so it is for jugaar. It includes keywords like maan_lo_ye, warna_maan_lo_ye, bas_maan_le_bhai for if else structure and ghuma_de for "for loop".
   My project includes 3 files.
   - **Scanner File:** jugr_scanner.l
   - **Test Program File:** check_prime_number.jugr
   - **Output File:** tokens_report.txt
3. **My Jugr Language Components:**
   i. **Keywords:**

Keyword	Token Type	Standard C/C++ Equivalent	Purpose
pura_number	KEYWORD	Int	Represents a whole number (integer).
thora_number	KEYWORD	float	Represents a floating-point number.
jumla	KEYWORD	string	Represents a sequence of characters (string).
harf	KEYWORD	char	Represents a single character.

Keyword	**Token Type**	**Standard C/C++ Equivalent**	**Purpose**
khali_kaam_kar	KEYWORD	void	Represents a function that returns no value.
shamil_karo	KEYWORD	#include	Preprocessor directive for including libraries.
shuru_hoja	KEYWORD	main	Entry point of the program.
maan_lo_ye	KEYWORD	If	Conditional branching keyword.
warna_maan_lo_ye	KEYWORD	else if	Secondary conditional check.
bas_maan_lo_bhai	KEYWORD	else	Default block if all conditions are false.
ghuma_de	KEYWORD	For	Loop initialization keyword.
ghumata_reh	KEYWORD	while	Loop execution keyword.
pehle_kar	KEYWORD	Do	Start of a do-while loop.
tham_ja	KEYWORD	break	Exits a loop immediately.
wapas_aja	KEYWORD	continue	Skips the current loop iteration.
nikal_lo	KEYWORD	return	Returns a value from a function.
sunle	KEYWORD	cin / scanf	Input operation.
likhle	KEYWORD	cout / printf	Output operation.

ii. Operators:

Operator	Token Type	Standard C/C++ Equivalent	Purpose
ke_barabar_ha	OPERATOR	==	Equality check.
ke_barabar_nahi_ha	OPERATOR	!=	Inequality check.
se_bada_ha	OPERATOR	>	Greater than.
se_chota_ha	OPERATOR	<	Less than.
se_bada_ya_barabar_ha	OPERATOR	>=	Greater than or equal to.
se_chota_ya_barabar_ha	OPERATOR	<=	Less than or equal to.
palat_de	OPERATOR	!	Logical NOT (negation).
aur_bhi	OPERATOR	&&	Logical AND.
ya_phir	OPERATOR	||	Loginal OR

Operator	Token Type	Purpose
=	OPERATOR	Assignment.
+, -	OPERATOR	Addition and Subtraction.
*, /	OPERATOR	Multiplication and Division.
%	OPERATOR	Modulo (Remainder).
++, --	OPERATOR	Increment and Decrement.

iii. Punctuators:

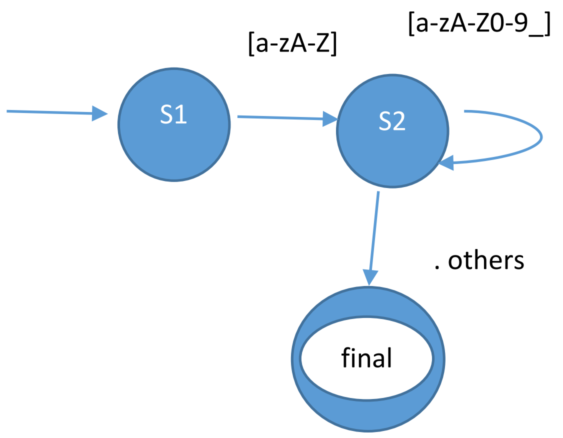
Element	Token Type	Purpose
;	TERMINATOR	Statement termination.
{, }	PUNCTUATION	Block start and end (scope definition).
(,)	PUNCTUATION	Function calls, argument lists, expression grouping.
[,]	PUNCTUATION	Array indexing.
,	PUNCTUATION	Separator for lists or arguments.

iv. Res:

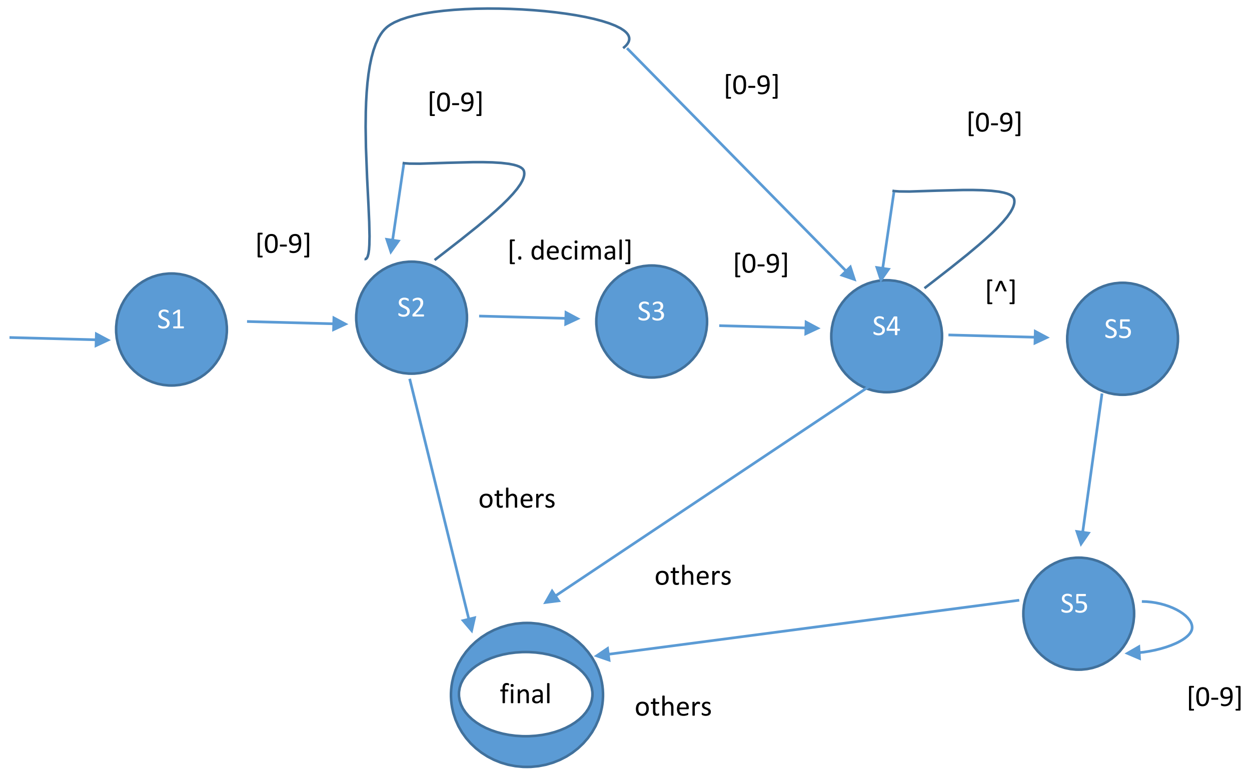
Token Type	Standard Regex	Flex Implementation
Keyword	[maan_lo_ye | warna_maan_lo_ye | etc]	"keyword" (Exact String Match)
Identifier	[a-zA-Z][_a-zA-Z0-9]*	[a-zA-Z][a-zA-Z0-9_]*
Number	[0-9]+(\.[0-9]+)?(\^[+-]?[0-9]+)?	[0-9]+(\.[0-9]+)?(\^[+-]?[0-9]+)?
String Literal	`"([^"\n]	.)*"`
Char Literal	`'([^'\n]	.)'`
Line Comment	//[^\n]*	"//"[^\n]*
Block Comment	`/*([^*]	*[^/])**/`
Whitespace	[\t\n]+	[\t\n]+

4. Finite Automatas:

Identifiers:



Numbers:



Commands To Run Project:

Here are the commands I run on my project to run:

- flex jugr_scanner.l
- gcc lex.yy.c -o scanner –lfl
- ./scanner < check_prime_number.jugr > tokens_report.txt

After running these are report is generated for tokens and errors records.