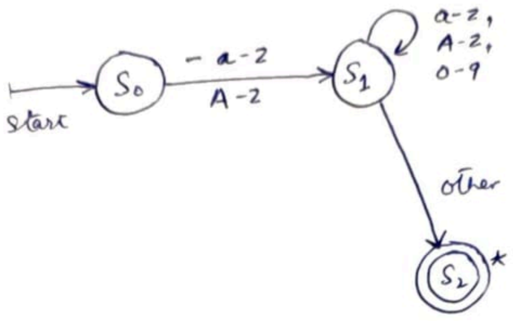
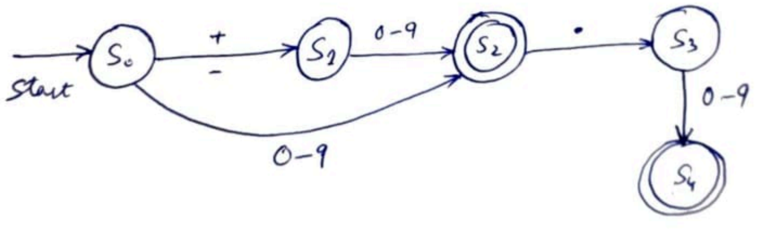
Token Type	Regular Expression (Regex)	Description
Identifier	[A-Za-z_][A-Za-z0-9_]*	Begins with a letter or underscore, followed by letters or digits.
Numbers	(+-)?[0-9]+[.[0-9]]?	All kind of Numbers
Integer Constant	[+-]?[0-9]+	One or more digits.
Float Constant	[0-9]*\.[0-9]+	Contains a decimal point with digits before and/or after.
Exponential Number	[0-9]+([eE][+-]?[0-9]+)	Integer or float followed by exponent (e.g., 1.2e+3).
String Literal	"[^"]*"	Double-quoted string containing any characters except ".
Whitespace / Newline	[\t\r\f]+ / \n	Spaces, tabs, or line breaks ignored by scanner.
Comment (Single-Line)	//[^\n]*	Everything from // to end of line.
Comment (Multi-Line)	`/*([^*]	*+[^/])*+/`
Keyword	`avn_(if	else
Operators (custom)	_+_, _-_, _*_, _/_, _++_, _--_	User-defined symbolic operators.
Operators (default)	=, \+, -, *, /, <, >, ==, !=, <<, >>	Standard Mini C++ arithmetic, relational, and stream operators.
Punctuations	_,_, ;’, ::, \(, \), \{, \}, ,	Commas, statement terminator, and grouping symbols.
Error / Unknown	.	Any single unmatched character.

Transition Diagram

- **Identifiers:**

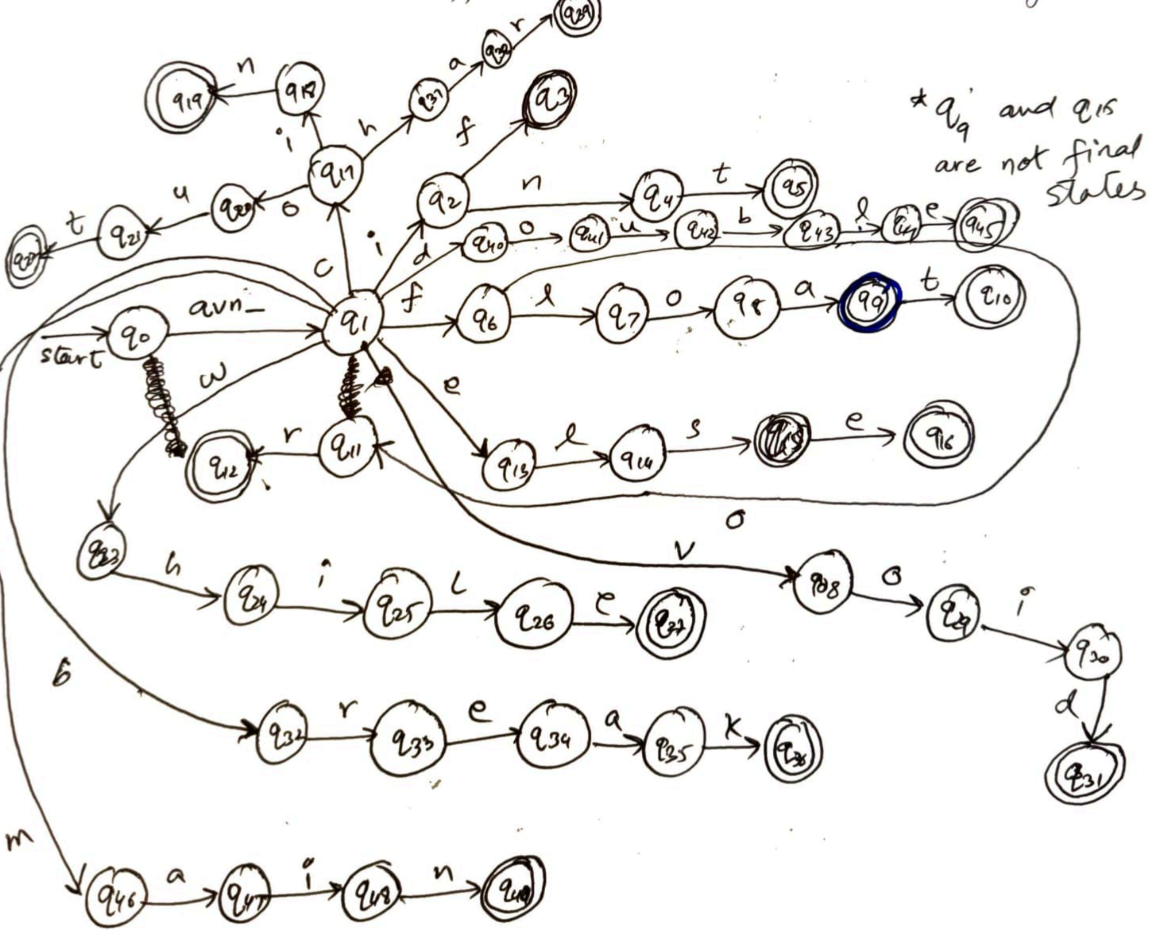


- **Numbers**

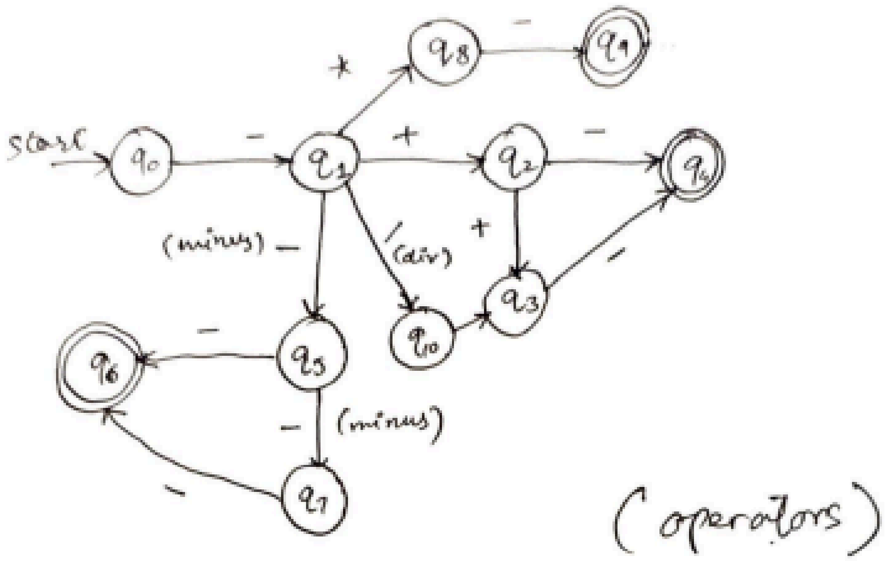


Finite Automata for all keywords

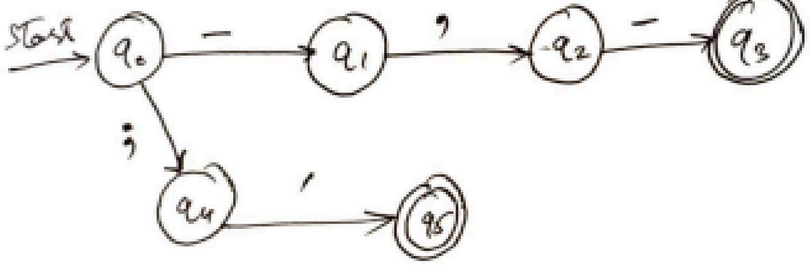
Since i'd have to design every FA separatley and its non-efficient, i'll draw it altogether.



Symbols and Punctuation:



(operators)



(punctations)

6 Exaplanations :

Avian is a mini version inspired by C++. I wonderd choosing a name with its own uniquness and came up with it to something as "birds can commicate to" even we can not understand them. Its structure is similar to C++. For keywords I added acronym for avian as "avn" with an underscore before every keyword for readability and memoization easy for me. And for the finite automatas, having same prefix was an approchable and affordable option. It helped with regular expression's straightforwadness too.