Name: M.Asaad
Roll No: L1F22BSCS0568

COMPILER CONSTRUCTION

Language Name: PakLang

OverView:

PakLang is a simplified Mini C++-like programming language designed for educational purposes.

Its goal is to help beginners understand lexical analysis through meaningful and easy-to-remember keywords derived from Urdu and simple English concepts.

PakLang supports:

- variables (numbers & decimals)
- keywords for decision control (agar, warna)
- loop control (dobarah, chalo, ruko)
- string printing (likho)
- boolean values (sach, jhoot)
- numeric operations
- punctuation controls similar to C-like languages

PakLang is intentionally minimalistic and readable.

Keywords:

Keyword	Meaning / Equivalent (in C/C++)	Usage Example
agar	`if`	`agar (x > 10)`
warna	`else`	`warna { ... }`
jab	`while`	`jab (i < 5)`
dobarah	`for`	`dobarah (i = 0; i < n; i = i + 1)`
wapis	`return`	`wapis x;`
karo	`do`	`karo { ... } jab (cond);`
chalo	`continue`	`chalo;`
ruko	`break`	`ruko;`

Keyword	Meaning / Equivalent (in C/C++)	Usage Example
`sach`	`true`	`bol flag = sach;`
`jhoot`	`false`	`bol flag = jhoot;`
`likho`	`print / output`	`likho("Hello");`
`riyazi`	`int`	`riyazi a = 10;`
`angerazi`	`float`	`angerazi b = 3.14;`
`poucho`	`input`	`poucho x;`
`HanYahNah`	`bool`	`HanYahNah ok = sach;`

Reason of choosing these keywords:

The keywords in **PakLang** were carefully selected to make programming feel familiar and intuitive for beginners, especially those who think and communicate in Urdu or Hindi. Each keyword is meaningful, short, and easy to pronounce — helping learners quickly connect the logic of programming with everyday language.

Here's the reasoning behind some of the core choices:

- **`agar` (if)** → In Urdu/Hindi, *agar* means "if" or "in case." It naturally expresses a condition, just like the English *if*, making it instantly understandable when writing conditional statements.
  Example: `agar (x > 10)` reads like "if x is greater than 10."
- **`warna` (else)** → Commonly used in everyday speech meaning "otherwise" or "else." It pairs naturally with *agar*, forming a clear and conversational logic structure — *agar... warna...* ("if... otherwise...")
- **`jab` (while)** → Literally means "when" or "while." It fits perfectly for loops that continue as long as a condition is true.
- **`dobarah` (for)** → Means "again" or "repeat." Ideal for representing *for loops*, where an action repeats multiple times.
- **`wapis` (return)** → The word *wapis* means "to return" or "to give back," exactly matching the concept of returning a value from a function.
- **`karo` (do)** → A simple word meaning "do" or "perform." It captures the directness of executing an action, as in *do-while* loops.
- **`chalo` (continue)** → Means "go on" or "keep going." Perfectly fits the *continue* keyword, telling the program to move on to the next iteration.
- **`ruko` (break)** → Means "stop" or "pause." This makes sense for breaking out of a loop or halting execution.
- **`sach` (true)** and **`jhoot` (false)** → Everyday words meaning "truth" and "falsehood." These feel much more relatable than *true* and *false*, making logical expressions easier to grasp.
- **`likho` (print/output)** → Literally means "write." Since printing or displaying output involves writing something on the screen, it's a natural, intuitive fit.

- **`riyazi` (integer)** → Comes from *riyaziyat* (mathematics). It represents numeric values (integers), directly connecting to basic arithmetic.
- **`angerazi` (float)** → A lighthearted, easy word representing decimal or non-integer numbers — keeping the tone friendly and simple.
- **`poucho` (input)** → Means "ask" or "inquire." This aligns perfectly with taking input from the user — the program "asks" for information.
- **`HanYahNah` (boolean type)** → Translates to "Yes or No," directly representing boolean logic (true/false). It's descriptive and easy for beginners to relate to.

Operators:

Operator	Meaning
`+`	Addition
`-`	Subtraction
`*`	Multiplication
`/`	Division
`==`	Equality Comparison

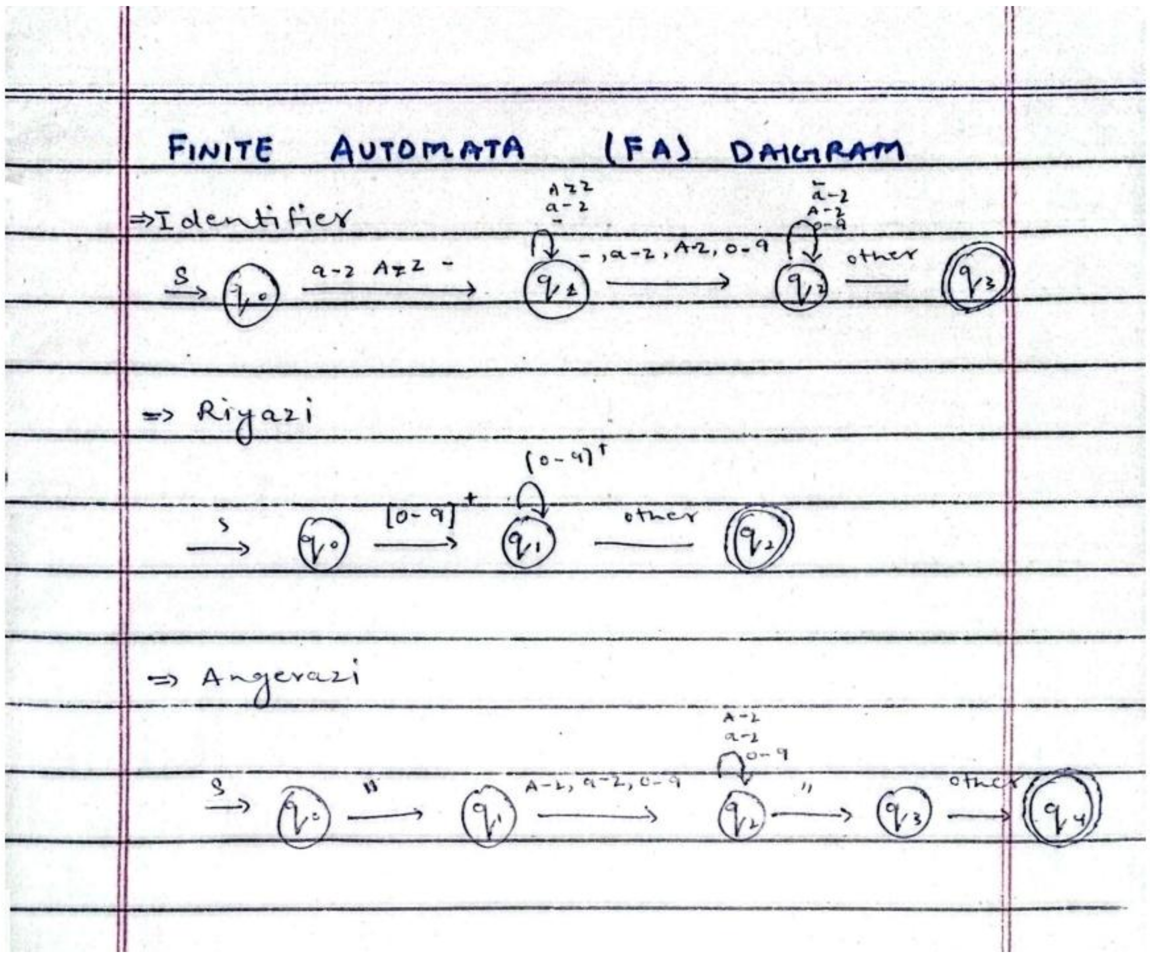
Punctuation:

Symbol	Usage
`( )`	Grouping
`{ }`	Code blocks
`,`	Separator
`;`	Statement terminator

Regular Expression:

Token Type	Regular Expression	Examples
Identifier	[A-Za-z_][A-Za-z0-9_]*	testVar, my_var, x1
Keyword	"agar" | "warna" | "jab" | "dobarah" | "wapis" | "karo" | "chalo" | "ruko" | "sach" | "jhoot" | "likho" | "riyazi" | "angerazi" | "poucho" | "HanYahNah"	agar, warna, HanYahNah
Integer Number	[0-9]+	50, 123, 999
Decimal Number	[0-9]+\.[0-9]+	10.75, 3.14, 0.5
String Literal	`"([^"\]	\.)*"`
Operators	`+	-
Punctuations	`(	)
Unknown Token	.	@, #, $

Finite Automata Diagram:



Explanation of Choices:

1. Keywords were chosen to be simple and intuitive.

2. Many Urdu-derived words increase readability.

3. Operators and punctuations follow C-style for familiarity.

4. Regex rules are chosen to be clear, standard, and accepted by Flex.

5. FAs are minimal by design and represent classic patterns for identifiers & numbers.

6. The language is unique, ensuring originality.

Repository Link:

https://github.com/Asaad-108/Paklang