

# Program-(6A)

Aim: write a program to perform given Prolog Examples.

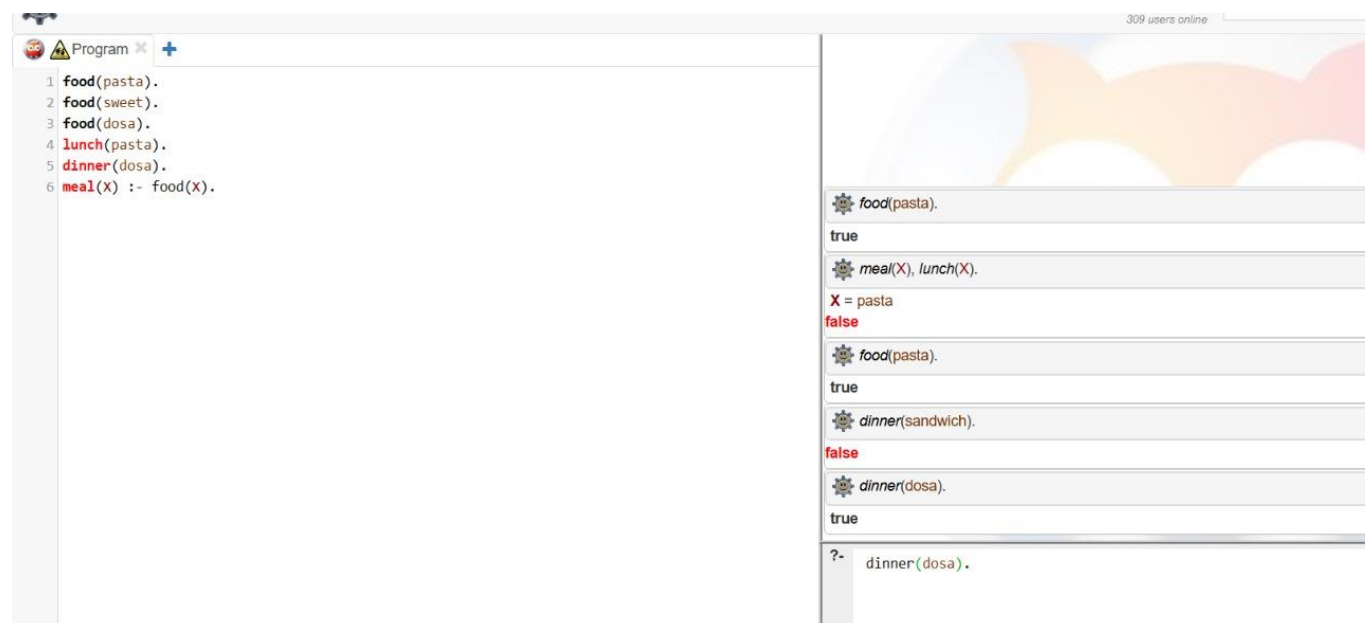
## Theory:

Prolog, a declarative programming language, operates on a foundation of logical inference and pattern matching. Facts and Rules: In Prolog, you define facts and rules. Facts represent ground truths, while rules establish relationships or conditions. Queries: Queries drive Prolog's execution. You pose queries to the system, which then attempts to find solutions based on the defined facts and rules. Variables: Prolog uses variables to enable flexible pattern matching and querying. Variables start with an uppercase letter or underscore.

## Algorithm:

Initialize Knowledge Base: Start with initializing the knowledge base by adding the facts about who studies which course and who teaches which course. Professor Inference Rule: Define the inference rule `professor(X, Y)` which states that a person X is a professor if they teach a course Z and a person Y studies that course Z. Querying: Once the knowledge base is set up and the inference rule is defined, you can query the system to find out who is a professor for a given course. Execution: During execution, Prolog will match the query against the defined facts and rules. It will find all instances where the `professor(X, Y)` rule is satisfied, i.e., where X teaches a course Z and Y studies that same course Z.

## Code:



The screenshot shows a Prolog IDE with a program editor on the left and a console on the right. The program editor contains the following code:

```
1 food(pasta).
2 food(sweet).
3 food(dosa).
4 lunch(pasta).
5 dinner(dosa).
6 meal(X) :- food(X).
```

The console on the right shows the execution of several queries:

- `food(pasta).` returns `true`.
- `meal(X), lunch(X).` returns `X = pasta` and `false`.
- `food(pasta).` returns `true`.
- `dinner(sandwich).` returns `false`.
- `dinner(dosa).` returns `true`.
- `?- dinner(dosa).` is shown at the bottom.

# Program-(6B)

Aim: write a program to perform given Prolog Examples.

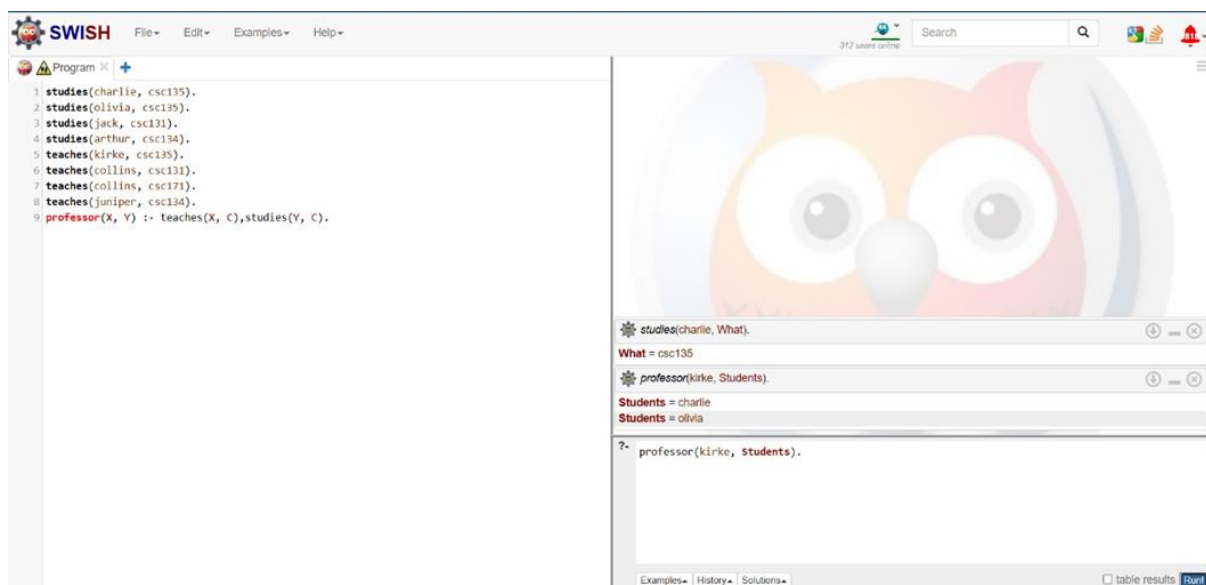
## Theory:

Prolog, a declarative programming language, operates on a foundation of logical inference and pattern matching. Facts and Rules: In Prolog, you define facts and rules. Facts represent ground truths, while rules establish relationships or conditions. Queries: Queries drive Prolog's execution. You pose queries to the system, which then attempts to find solutions based on the defined facts and rules. Variables: Prolog uses variables to enable flexible pattern matching and querying. Variables start with an uppercase letter or underscore.

## Algorithm:

Initialize Knowledge Base: Start with initializing the knowledge base by adding the facts about who studies which course and who teaches which course. Professor Inference Rule: Define the inference rule `professor(X, Y)` which states that a person X is a professor if they teach a course Z and a person Y studies that course Z. Querying: Once the knowledge base is set up and the inference rule is defined, you can query the system to find out who is a professor for a given course. Execution: During execution, Prolog will match the query against the defined facts and rules. It will find all instances where the `professor(X, Y)` rule is satisfied, i.e., where X teaches a course Z and Y studies that same course Z.

## Code:



The screenshot shows the SWISH Prolog environment. The left pane contains the following Prolog code:

```
1 studies(charlie, csc135).
2 studies(olivia, csc135).
3 studies(jack, csc131).
4 studies(arthur, csc134).
5 teaches(kirke, csc135).
6 teaches(collins, csc131).
7 teaches(collins, csc171).
8 teaches(juniper, csc134).
9 professor(X, Y) :- teaches(X, C), studies(Y, C).
```

The right pane shows the execution results for the query `studies(charlie, What).` and `professor(kirke, Students).`

```
studies(charlie, What).
What = csc135.

professor(kirke, Students).
Students = charlie
Students = olivia

?- professor(kirke, Students).
```

At the bottom, there are tabs for Examples, History, and Solutions, and a checkbox for "table results" with a "Run" button.