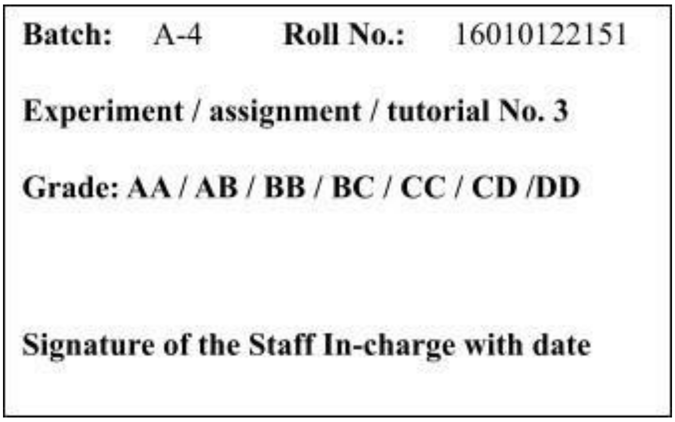
K. J. Somaiya College of Engineering, Mumbai-77



Batch: A-4 **Roll No.:** 16010122151

Experiment / assignment / tutorial No. 3

Grade: AA / AB / BB / BC / CC / CD /DD

Signature of the Staff In-charge with date

Title: Implementation of Goal based agent architecture using PROLOG.

Objective: To use the concepts of knowledge engineering to design and solve moderate complex problem.

Expected Outcome of Experiment:

Course Outcome	After successful completion of the course students should be able to
CO1	Design AI solution with appropriate choice of agent architecture
CO3	Represent and formulate the knowledge to solve the problems using various reasoning techniques

Books/ Journals/ Websites referred:

1. **https://www.csupomona.edu/~jrfisher/www/prolog_tutorial/contents.html**
2. **http://www.csupomona.edu/~jrfisher/www/prolog_tutorial/pt_framer.html**
3. **http://www.doc.gold.ac.uk/~mas02gw/prolog_tutorial/prologpages/**
4. **http://classes.soe.ucsc.edu/cmps112/Spring03/languages/prolog/PrologIntro.pdf**
5. **“Prolog: Programming for Artificial Intelligence” by Ivan Bratko, Pearson education Publications**
6. **“Artificial Intelligence: a Modern Approach” by Russel and Norving, Pearson education Publications**
7. **“Artificial Intelligence” By Rich and knight, Tata Mcgraw Hill Publications**

Pre Lab/ Prior Concepts:

Agents, Agent Architecture, Programming with PROLOG

Historical Profile:

Knowledge is vast, uncertain and continuously changing. These properties of knowledge make it difficult to arrive at a result. A murder mystery is a kind of situation which depicts the uncertain nature of knowledge and also emphasizes the need of choosing right clauses from entire knowledgebase to make a decision. He goal based agent architecture and some knowledge engineering can help in solutioning of such problems.

The logical agents are complex but they can reason and learn from the actions and new precepts. They are less like acting and think like humans but more like acting and thinking rational agents.

Knowledge and reasoning play a crucial role in dealing with partially observable environments. A knowledge based agent can combine the general knowledge with current percept to infer the hidden aspects of the current state prior to selecting actions.

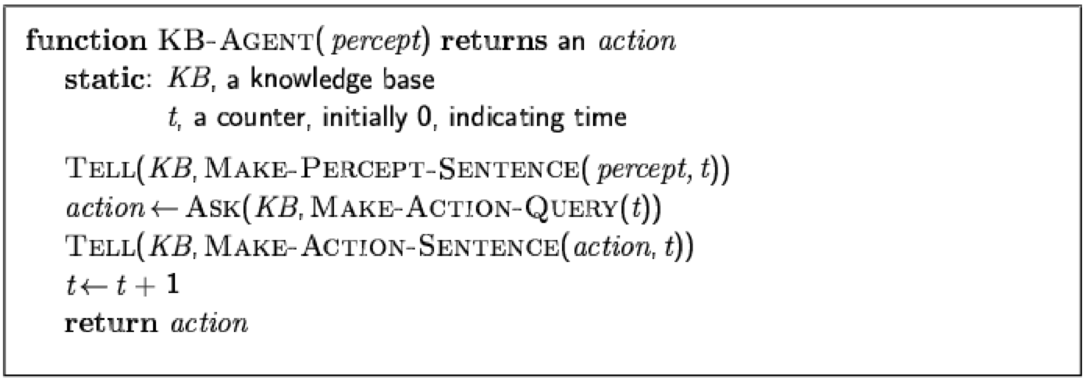
New Concepts to be learned:

Knowledge engineering, implementing complex agent architecture, uncertainty in knowledge.

The Knowledge Engineering Process

1. Identify the task
2. Assemble the relevant knowledge
3. Decide on vocabulary of predicates, functions and constants
4. Encode general knowledge about the domain
5. Encode description of specific problem instance
6. Pose queries to the inference procedure and get answers
7. Debug the knowledge base

Algorithm for KB-Agent:



```
function KB-AGENT(percept) returns an action
    static: KB, a knowledge base
            t, a counter, initially 0, indicating time

    TELL(KB, MAKE-PERCEPT-SENTENCE(percept, t))
    action ← ASK(KB, MAKE-ACTION-QUERY(t))
    TELL(KB, MAKE-ACTION-SENTENCE(action, t))
    t ← t + 1
    return action
```

Applications of Knowledge Engineering Steps to the Problem

Knowledge engineering involves building intelligent systems that simulate human reasoning. The steps applied to your murder mystery case:

1. Problem Definition
   - *What happened?* Henry Lancaster was found murdered.
   - *Goal:* Identify the murderer using logical deduction.
   - *Domain:* A locked mansion with 5 suspects and multiple clues.
2. Knowledge Acquisition
   - Information gathered:
     - Background of suspects
     - Motives
     - Alibis
     - Physical evidence
     - Timeline clues
3. Knowledge Representation
   - Represented using facts (e.g., suspect(X)), and rules (e.g., murderer(X) :- ...) in PROLOG.
   - Facts and rules were structured to allow deductive reasoning.
4. Inference/Reasoning
   - Implemented with goal-based rules:
     - Who could be a possible suspect?
     - Who is linked to strong evidence?
     - Who fits the criteria of the murderer?
5. Validation & Explanation
   - Each step in the agent's deduction can be traced via PROLOG queries.
   - The solution is explainable by viewing rule triggers and fact matches.

Solving Using Goal-Based Architecture

A Goal-Based Agent tries to achieve a specific goal using a sequence of rules. Here, the goal is to identify the murderer.

- Initial State: A murder occurred with 5 suspects and 5 pieces of evidence.
- Goal State: Determine the murderer.

- Rules (Action): Deduce using logic:
  - Determine who had motive + weak/questionable alibi.
  - Cross-reference with evidence.
  - Final deduction based on time of activity and physical trace.

General Rules

```
% General Rule 1: A suspect is someone present during the crime
suspect(X) :- person(X), was_present(X).

% General Rule 2: A person with motive and no solid alibi is suspicious
possible_suspect(X) :- suspect(X), motive(X, _), (alibi(X, weak); alibi(X, questionable)).

% General Rule 3: A prime suspect is a possible suspect linked to physical evidence
prime_suspect(X) :- possible_suspect(X), evidence_linked_to(X).

% General Rule 4: The murderer fits multiple evidence and behavioral clues
murderer(X) :- prime_suspect(X), matches_crime_conditions(X).
```

Problem-Specific Rules

```
% Problem-specific Rule 1: Motives
motive(margaret_lancaster, inheritance).
motive(charles_lancaster, revenge).
motive(dr_walter_graham, cover_up).
motive(sophia_carter, jealousy).
motive(james_holloway, financial_gain).

% Problem-specific Rule 2: Evidence presence
evidence(gunshot_residue, gloves).
evidence(burnt_will, fireplace).
evidence(bloody_handprint, desk).
evidence(hidden_gun, garden).
evidence(security_log, door_opened_at_midnight).

% Rule to determine who hid the gun
hid_weapon(X) :- suspect(X), evidence(hidden_gun, garden).

% Final rule: Identifying murderer based on timing and residue
murderer(X) :-
    prime_suspect(X),
    evidence(gunshot_residue, gloves),
    evidence(security_log, door_opened_at_midnight).
```

Problem Statement:

The Silent Witness

Detective **Alexander Pierce** was called to **Ravenswood Mansion**, where a **wealthy philanthropist, Henry Lancaster,** had been found dead in his library. The scene was eerie—Henry was slumped over his desk, a **single gunshot wound** to his head.

The strange thing? **The gun was missing**. The mansion doors were locked, and only **five people** were inside that night. Someone in the house had killed Henry, but **who and why?**

Suspects:

1. **Margaret Lancaster** - The victim’s wife, set to inherit his wealth.
2. **Charles Lancaster** - The estranged son, angry about being removed from the will.
3. **Dr. Walter Graham** - The family doctor, who knew Henry’s medical history.
4. **Sophia Carter** - The housekeeper, who was secretly in love with Henry.
5. **James Holloway** - The business partner, who had financial disputes with Henry.

Clues Found:

- A **gunshot residue stain** on someone's gloves.
- A **half-burnt will** in the fireplace.
- A **hidden gun** found in the garden.
- A **bloody handprint** on the victim’s desk.
- A **security log** that showed a door opened briefly at midnight.

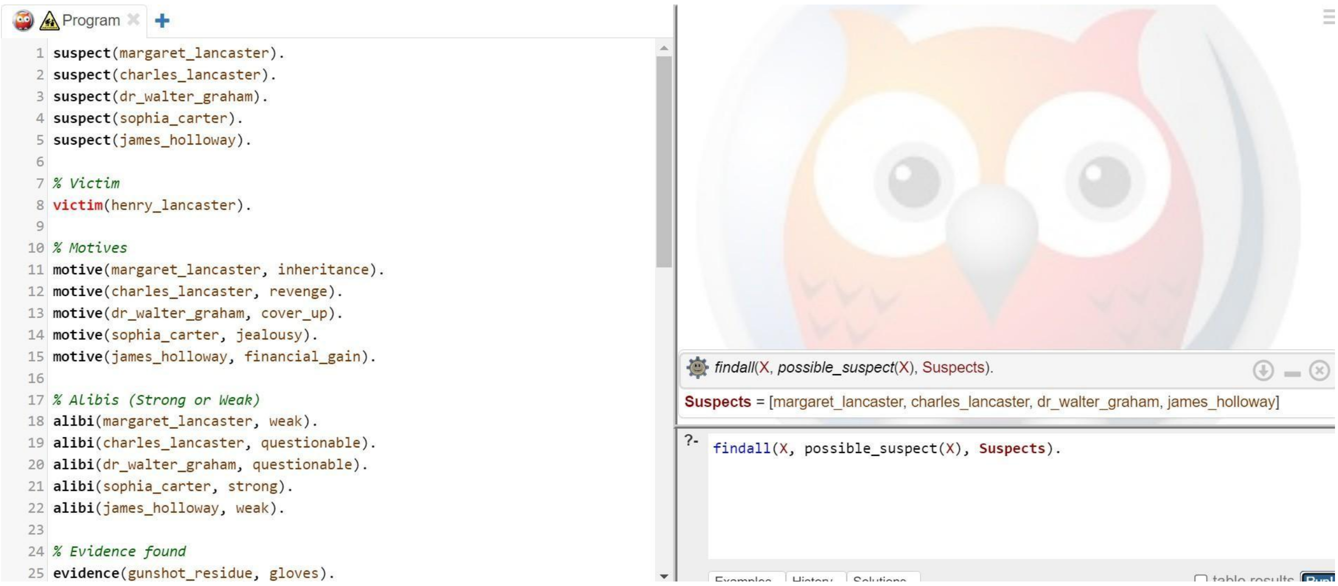
The mansion's **only witness** was the family **parrot, Jasper**, who kept repeating, *"Bang! Bang! He said ‘No!’"*

Code:-

```
% Suspects

suspect(margaret_lancaster).

suspect(charles_lancaster).

suspect(dr_walter_graham).

suspect(sophia_carter).

suspect(james_holloway).
```

```
% Victim
victim(henry_lancaster).

% Motives
motive(margaret_lancaster, inheritance).
motive(charles_lancaster, revenge).
motive(dr_walter_graham, cover_up).
motive(sophia_carter, jealousy).
motive(james_holloway, financial_gain).

% Alibis (Strong or Weak)
alibi(margaret_lancaster, weak).
alibi(charles_lancaster, questionable).
alibi(dr_walter_graham, questionable).
alibi(sophia_carter, strong).
alibi(james_holloway, weak).

% Evidence found
evidence(gunshot_residue, gloves).
evidence(burnt_will, fireplace).
evidence(hidden_gun, garden).
evidence(bloody_handprint, desk).
evidence(security_log, door_opened_at_midnight).

% Rules

% 1. A person with motive and weak alibi is a possible suspect.
possible_suspect(X) :-
    suspect(X),
    motive(X, _),
    (alibi(X, weak); alibi(X, questionable)).

% 2. A person linked to crucial evidence becomes a prime suspect.
prime_suspect(X) :-
    possible_suspect(X),
    (evidence(gunshot_residue, gloves);
    evidence(burnt_will, fireplace);
    evidence(bloody_handprint, desk)).

% 3. The real murderer had gunshot residue and was awake at midnight.
murderer(X) :-
    prime_suspect(X),
    evidence(gunshot_residue, gloves),
    evidence(security_log, door_opened_at_midnight).

% 4. The missing gun was found in the garden—only someone who went outside could have hidden it.
hid_weapon(X) :-
    suspect(X),
    evidence(hidden_gun, garden).
```

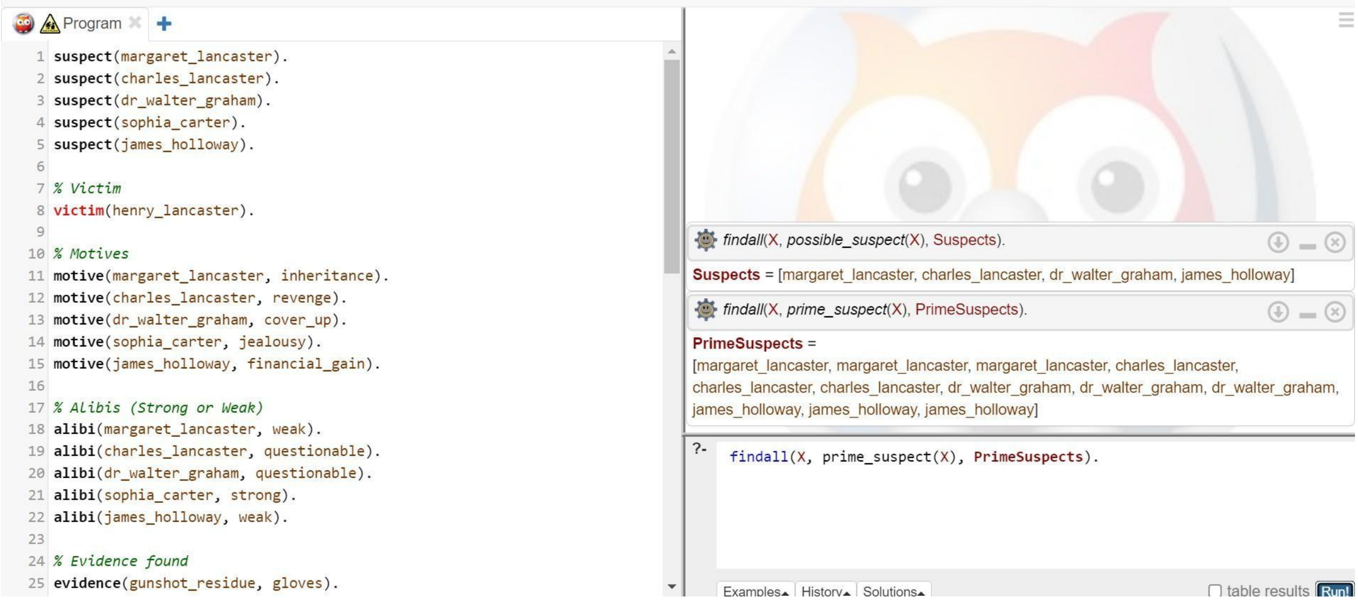
Output:-



```
suspect(margaret_lancaster).
suspect(charles_lancaster).
suspect(dr_walter_graham).
suspect(sophia_carter).
suspect(james_holloway).

% Victim
victim(henry_lancaster).

% Motives
motive(margaret_lancaster, inheritance).
motive(charles_lancaster, revenge).
motive(dr_walter_graham, cover_up).
motive(sophia_carter, jealousy).
motive(james_holloway, financial_gain).

% Alibis (Strong or Weak)
alibi(margaret_lancaster, weak).
alibi(charles_lancaster, questionable).
alibi(dr_walter_graham, questionable).
alibi(sophia_carter, strong).
alibi(james_holloway, weak).

% Evidence found
evidence(gunshot_residue, gloves).
```

```
findall(X, possible_suspect(X), Suspects).
Suspects = [margaret_lancaster, charles_lancaster, dr_walter_graham, james_holloway]
```

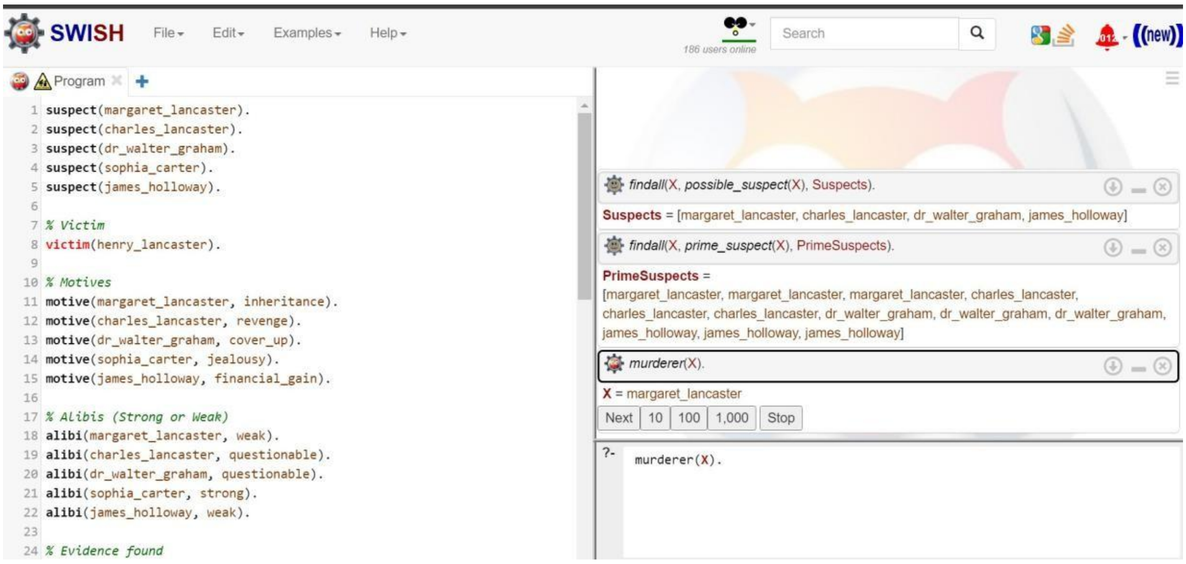
```
?- findall(X, possible_suspect(X), Suspects).
```



```
suspect(margaret_lancaster).
suspect(charles_lancaster).
suspect(dr_walter_graham).
suspect(sophia_carter).
suspect(james_holloway).

% Victim
victim(henry_lancaster).

% Motives
motive(margaret_lancaster, inheritance).
motive(charles_lancaster, revenge).
motive(dr_walter_graham, cover_up).
motive(sophia_carter, jealousy).
motive(james_holloway, financial_gain).

% Alibis (Strong or Weak)
alibi(margaret_lancaster, weak).
alibi(charles_lancaster, questionable).
alibi(dr_walter_graham, questionable).
alibi(sophia_carter, strong).
alibi(james_holloway, weak).

% Evidence found
evidence(gunshot_residue, gloves).
```

```
findall(X, possible_suspect(X), Suspects).
Suspects = [margaret_lancaster, charles_lancaster, dr_walter_graham, james_holloway]

findall(X, prime_suspect(X), PrimeSuspects).
PrimeSuspects =
[margaret_lancaster, margaret_lancaster, margaret_lancaster, charles_lancaster,
charles_lancaster, charles_lancaster, dr_walter_graham, dr_walter_graham, dr_walter_graham,
james_holloway, james_holloway, james_holloway]
```

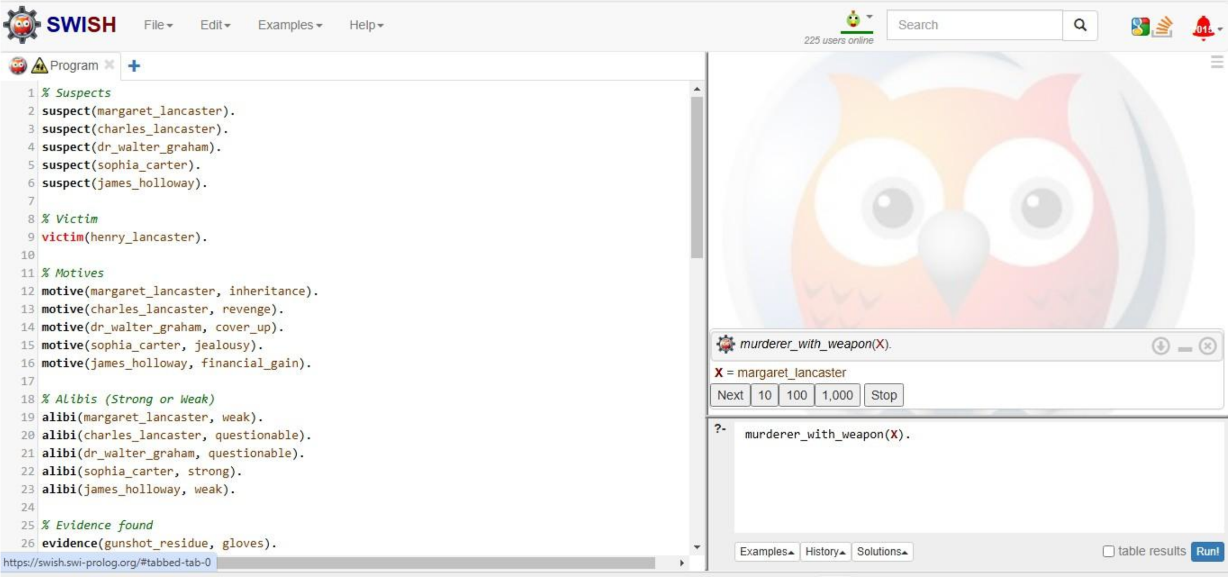
```
?- findall(X, prime_suspect(X), PrimeSuspects).
```



```
suspect(margaret_lancaster).
suspect(charles_lancaster).
suspect(dr_walter_graham).
suspect(sophia_carter).
suspect(james_holloway).

% Victim
victim(henry_lancaster).

% Motives
motive(margaret_lancaster, inheritance).
motive(charles_lancaster, revenge).
motive(dr_walter_graham, cover_up).
motive(sophia_carter, jealousy).
motive(james_holloway, financial_gain).

% Alibis (Strong or Weak)
alibi(margaret_lancaster, weak).
alibi(charles_lancaster, questionable).
alibi(dr_walter_graham, questionable).
alibi(sophia_carter, strong).
alibi(james_holloway, weak).

% Evidence found
```

```
findall(X, possible_suspect(X), Suspects).
Suspects = [margaret_lancaster, charles_lancaster, dr_walter_graham, james_holloway]

findall(X, prime_suspect(X), PrimeSuspects).
PrimeSuspects =
[margaret_lancaster, margaret_lancaster, margaret_lancaster, charles_lancaster,
charles_lancaster, charles_lancaster, dr_walter_graham, dr_walter_graham, dr_walter_graham,
james_holloway, james_holloway, james_holloway]

murderer(X).
X = margaret_lancaster
```

```
?- murderer(X).
```

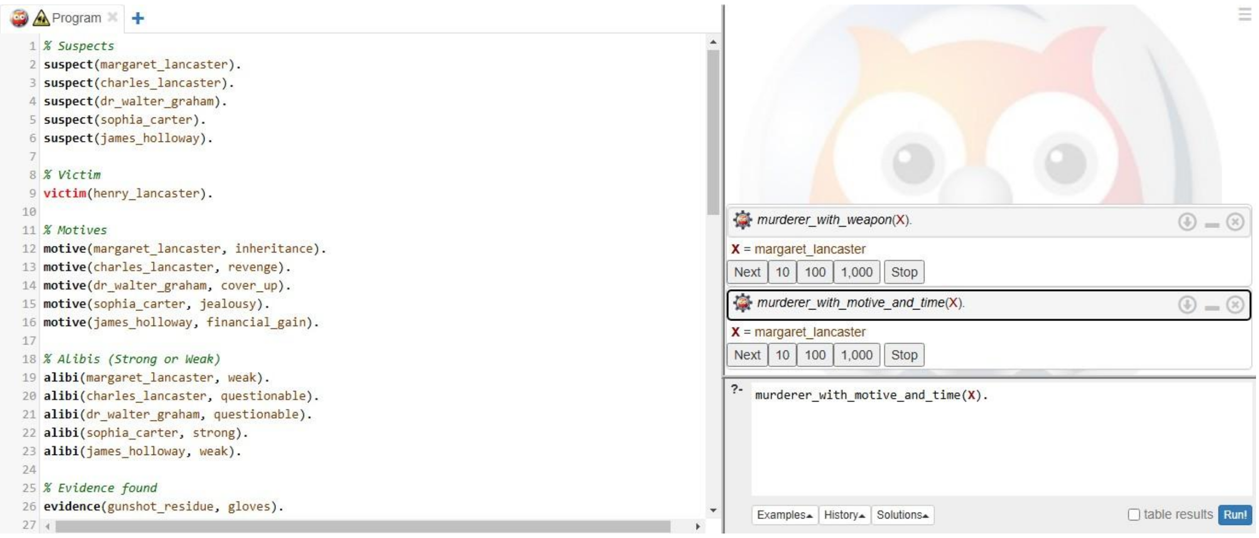


```
% Suspects
suspect(margaret_lancaster).
suspect(charles_lancaster).
suspect(dr_walter_graham).
suspect(sophia_carter).
suspect(james_holloway).

% Victim
victim(henry_lancaster).

% Motives
motive(margaret_lancaster, inheritance).
motive(charles_lancaster, revenge).
motive(dr_walter_graham, cover_up).
motive(sophia_carter, jealousy).
motive(james_holloway, financial_gain).

% Alibis (Strong or Weak)
alibi(margaret_lancaster, weak).
alibi(charles_lancaster, questionable).
alibi(dr_walter_graham, questionable).
alibi(sophia_carter, strong).
alibi(james_holloway, weak).

% Evidence found
evidence(gunshot_residue, gloves).
```

https://swish.swi-prolog.org/#tabbed-tab-0

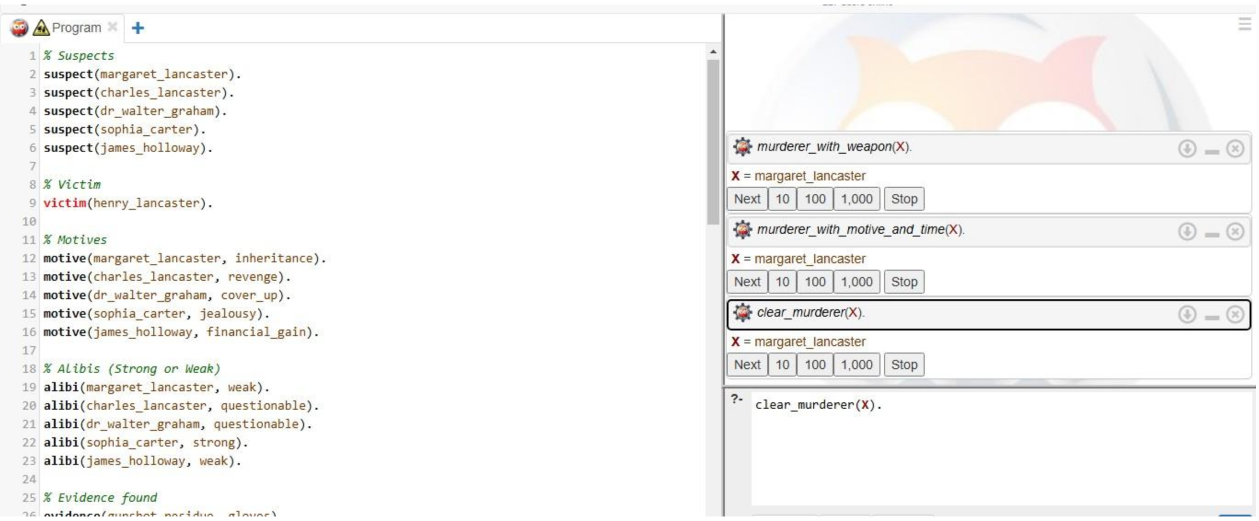
```
murderer_with_weapon(X).
X = margaret_lancaster
```

```
?- murderer_with_weapon(X).
```



```
% Suspects
suspect(margaret_lancaster).
suspect(charles_lancaster).
suspect(dr_walter_graham).
suspect(sophia_carter).
suspect(james_holloway).

% Victim
victim(henry_lancaster).

% Motives
motive(margaret_lancaster, inheritance).
motive(charles_lancaster, revenge).
motive(dr_walter_graham, cover_up).
motive(sophia_carter, jealousy).
motive(james_holloway, financial_gain).

% Alibis (Strong or Weak)
alibi(margaret_lancaster, weak).
alibi(charles_lancaster, questionable).
alibi(dr_walter_graham, questionable).
alibi(sophia_carter, strong).
alibi(james_holloway, weak).

% Evidence found
evidence(gunshot_residue, gloves).
```

```
murderer_with_weapon(X).
X = margaret_lancaster

murderer_with_motive_and_time(X).
X = margaret_lancaster

?- murderer_with_motive_and_time(X).
```



```
% Suspects
suspect(margaret_lancaster).
suspect(charles_lancaster).
suspect(dr_walter_graham).
suspect(sophia_carter).
suspect(james_holloway).

% Victim
victim(henry_lancaster).

% Motives
motive(margaret_lancaster, inheritance).
motive(charles_lancaster, revenge).
motive(dr_walter_graham, cover_up).
motive(sophia_carter, jealousy).
motive(james_holloway, financial_gain).

% Alibis (Strong or Weak)
alibi(margaret_lancaster, weak).
alibi(charles_lancaster, questionable).
alibi(dr_walter_graham, questionable).
alibi(sophia_carter, strong).
alibi(james_holloway, weak).

% Evidence found
evidence(gunshot_residue, gloves).
```

```
murderer_with_weapon(X).
X = margaret_lancaster

murderer_with_motive_and_time(X).
X = margaret_lancaster

clear_murderer(X).
X = margaret_lancaster

?- clear_murderer(X).
```

Team Members:

1. Hyder Presswala [16010122151**]**

2. Riddhi Patil [16010122138]

Post Lab Objective Questions

1. Which is not a Goal-based agent?

a. Inference

b. Search

c. Planning
d. Conclusion
e. Dynamic search.

Answer: Conclusion(d)

2. Which were built in such a way that humans had to supply the inputs and interpret the outputs?

a. Agents
b. Sensor
c. AI System
d. Actuators

Answer: AI System(c)

Post Lab Subjective Questions

Explain the role of PEAS and task environment in choosing the agent architecture. Justify your answer with an example.

PEAS (Performance measure, Environment, Actuators, Sensors) is a framework used to define an agent’s task environment. It helps in structuring the problem by identifying key elements necessary for the agent's design.

- **Performance Measure**: Defines the criteria for evaluating how well the agent is performing its task.
- **Environment**: The external context or surroundings within which the agent operates.
- **Actuators**: The components responsible for carrying out the agent’s actions in the environment.
- **Sensors**: The components that allow the agent to perceive or gather information from the environment.

Task Environment refers to the problem or situation the agent is designed to address. It includes everything the agent interacts with, such as the physical or virtual environment, tasks to be performed, and the constraints the agent must work under.

Choosing the agent architecture depends on the combination of PEAS and the task environment. For example:

- If the task environment involves simple, well-defined actions (e.g., a chess-playing agent), a **model-based reflex agent** or a **goal-based agent** may be appropriate, where performance measures like winning the game drive the architecture design.
- For an autonomous vehicle, the **task environment** is dynamic and complex, with tasks such as navigation, detecting obstacles, and responding to traffic signals. A **deliberative agent** with sensors for vision and actuators for steering, braking, and acceleration would be chosen, as its performance depends on real-time input, actions, and ongoing learning.

In both cases, PEAS helps in tailoring the architecture to the environment's requirements and the desired performance.