Vishwakarma Institute of Technology
Department of Computer Science Engineering (CS)

Name	Vaishnavi Vijay Arthamwar
Division	CS-A
Batch	B3
Roll No.	16
PRN No.	12220242
Department	Computer
Subject	Artificial Intelligence
Assignment No.	6

Assingment No. 6

Title: Implementation Expert system using PROLOG.

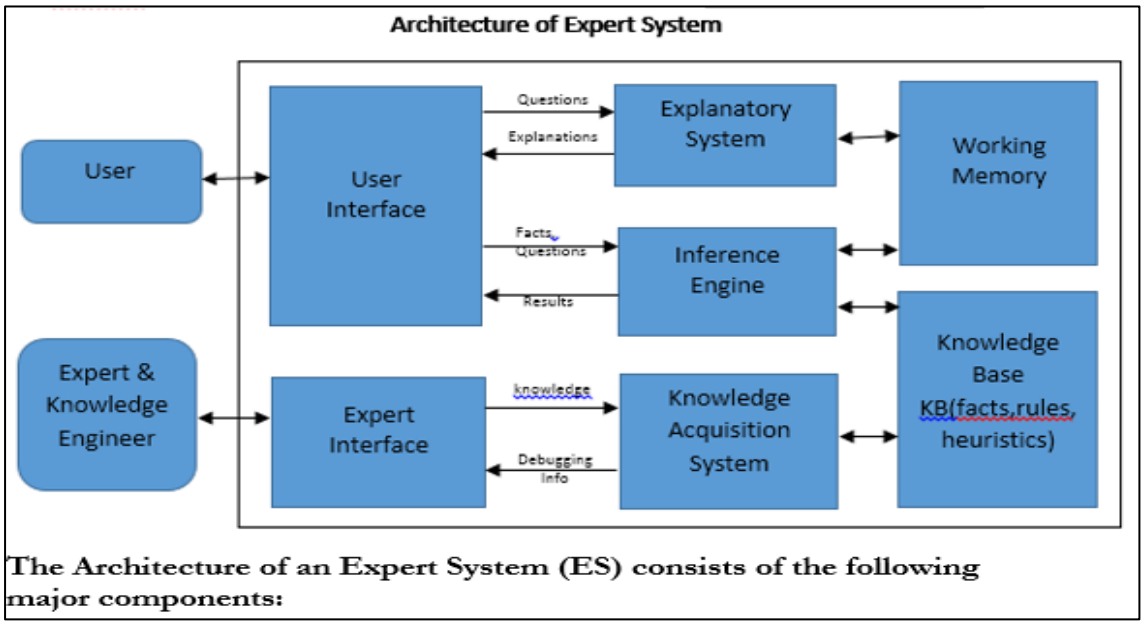
Theory:

1. **Predicate Logic:** Prolog is based on predicate logic, a formal system for expressing knowledge about the world using logical predicates and rules. In predicate logic, statements are represented as predicates, which are relationships between objects, and logical connectives such as AND, OR, and NOT are used to combine predicates.
2. **Knowledge Representation:** In Prolog, knowledge is represented using facts and rules. Facts are statements about the world that are assumed to be true, while rules define relationships and infer new information based on existing facts. Facts and rules are written in the form of predicates and clauses.
3. **Inference Engine:** Prolog's inference engine employs a process called resolution to derive new facts or verify the validity of queries based on the available knowledge base. It uses backtracking and unification to explore possible solutions to queries by matching them against the rules and facts in the knowledge base.
4. **Expert System Architecture:** Expert systems typically consist of three main components: a knowledge base, an inference engine, and a user interface. The knowledge base stores domain-specific information in the form of facts and rules. The inference engine uses logical reasoning to draw conclusions and provide advice based on the knowledge base. The user interface enables users to interact with the system by posing queries and receiving responses.
5. **Knowledge Representation in Prolog:** Prolog uses a declarative style of programming, where knowledge is represented as a collection of logical facts and rules. Facts represent basic assertions about the world, while rules define logical relationships and can infer new facts from existing ones. Prolog's syntax allows for the representation of complex relationships and hierarchical structures, making it suitable for expressing domain knowledge in a natural and concise manner.
6. **Logical Inference in Prolog:** Prolog's inference engine employs a resolution-based theorem proving mechanism to derive conclusions from the knowledge base. The process involves recursively applying rules and facts to satisfy user queries or goals. Prolog uses a depth-first search strategy combined with backtracking to explore possible solutions, making it efficient for reasoning with large knowledge bases. Additionally, Prolog utilizes unification, a process of finding variable bindings that make logical expressions equivalent, to match query patterns with facts and rules in the knowledge base.
7. **Rule-Based Reasoning:** Expert systems often rely on rule-based reasoning to make decisions or provide recommendations. In Prolog, rules are defined using Horn clauses,

which consist of a head (conclusion) and a body (conditions). When a query matches the head of a rule, Prolog attempts to satisfy the conditions in the body by unifying them with available facts in the knowledge base. If successful, the rule is considered applicable, and the inference engine proceeds to infer the conclusion stated in the head of the rule.

8. **Handling Uncertainty and Fuzzy Logic:** While Prolog excels at reasoning with deterministic knowledge, expert systems may encounter situations where uncertainty or ambiguity exists. To handle uncertainty, Prolog can be extended with probabilistic or fuzzy logic extensions. Probabilistic Prolog adds support for probabilistic inference, allowing for reasoning under uncertainty by assigning probabilities to facts and rules. Fuzzy logic extends Prolog's ability to deal with imprecise or vague information by using fuzzy sets and fuzzy rules, enabling more flexible reasoning in domains where precise measurements are difficult to obtain.
9. **Integration with External Systems:** Expert systems implemented in Prolog can be integrated with external systems and databases to enhance their functionality. Prolog provides mechanisms for interfacing with other programming languages and external data sources, allowing expert systems to access real-time information, perform complex computations, or interact with external services. This integration capability enables expert systems to leverage a wide range of resources and extend their reach beyond the limitations of the Prolog environment.
10. **Maintenance and Evolution:** Expert systems are not static entities; they evolve over time as new knowledge becomes available or domain requirements change. Prolog's modular and extensible nature facilitates the maintenance and evolution of expert systems by allowing incremental updates to the knowledge base and inference rules. Additionally, Prolog's rule-based approach promotes transparency and traceability, making it easier to understand and modify the system's behavior as needed.

Expert System Architecture:



The Architecture of an Expert System (ES) consists of the following major components:

Usecase: **Medical Diagnosis : The medical expert system will tell the patient the type of disease he/she has on the basis of the symptoms given by the patient.**

Code:

```
symptom('Flu').
symptom('Yellowish eyes and skin').
symptom('Dark color urine').
symptom('Pale bowel movement').
symptom('Fatigue').
symptom('Vomitting').
symptom('Fever').
symptom('Pain in joints').
symptom('Weakness').
symptom('Stomach Pain').

treatment('Flu', 'Drink hot water, avoid cold eatables.').
treatment('Yellowish eyes and skin', 'Put eye drops, have healthy sleep, do
not strain your eyes.').
treatment('Dark color urine', 'Drink lots of water, juices and eat fruits.
Avoid alcohol consumption.').
treatment('Pale bowel movement', 'Drink lots of water and exercise
regularly.').
treatment('Fatigue', 'Drink lots of water, juices and eat fruits.').
treatment('Vomitting', 'Drink salt and water.').
treatment('Fever', 'Put hot water cloth on head and take crocin.').
treatment('Pain in Joints', 'Apply pain killer and take crocin.').
treatment('Weakness', 'Drink salt and water, eat fruits.').
treatment('Stomach Pain', 'Avoid outside food and eat fruits.').

input :- dynamic(patient/2),
    repeat,
    symptom(X),
    write('Does the patient have '),
    write(X),
    write('? '),
    read(Y),
    assert(patient(X,Y)),
    \+ not(X='Stomach Pain'),
    not(output).

disease(hemochromatosis) :-
    patient('Stomach Pain',yes),
    patient('Pain in joints',yes),
    patient('Weakness',yes),
    patient('Dark color urine',yes),
    patient('Yellowish eyes and skin',yes).
```

```
disease(hepatitis_c) :-
    not(disease(hemochromatosis)),
    patient('Pain in joints',yes),
    patient('Fever',yes),
    patient('Fatigue',yes),
    patient('Vomitting',yes),
    patient('Pale bowel movement',yes).

disease(hepatitis_b) :-
    not(disease(hemochromatosis)),
    not(disease(hepatitis_c)),
    patient('Pale bowel movement',yes),
    patient('Dark color urine',yes),
    patient('Yellowish eyes and skin',yes).

disease(hepatitis_a) :-
    not(disease(hemochromatosis)),
    not(disease(hepatitis_c)),
    not(disease(hepatitis_b)),
    patient('Flu',yes),
    patient('Yellowish eyes and skin',yes).

disease(jaundice) :-
    not(disease(hemochromatosis)),
    not(disease(hepatitis_c)),
    not(disease(hepatitis_b)),
    not(disease(hepatitis_a)),
    patient('Yellowish eyes and skin',yes).

disease(flu) :-
    not(disease(hemochromatosis)),
    not(disease(hepatitis_c)),
    not(disease(hepatitis_b)),
    not(disease(hepatitis_a)),
    patient('Flu',yes).

output:-
    nl,
    possible_diseases,
    nl,
    advice.

possible_diseases :- disease(X), write('The patient may suffer from '),
write(X),nl.
advice :- symptom(X), patient(X,yes), treatment(X,Y), write(Y),nl, \+
not(X='Stomach Pain').
```

Output:

```
PS D:\sem2-2023\AI\practical6> swipl
Welcome to SWI-Prolog (threaded, 64 bits, version 9.2.2)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

1 ?- consult('expert.pl').
true.

2 ?- input.
Does the patient have Flu? yes
.
Does the patient have Yellowish eyes and skin? |: no.
Does the patient have Dark color urine? |: no.
Does the patient have Pale bowel movement? |: yes.
Does the patient have Fatigue? |: no.
Does the patient have Vomitting? |: yes.
Does the patient have Fever? |: yes.
Does the patient have Pain in joints? |: no.
Does the patient have Weakness? |: yes.
Does the patient have Stomach Pain? |: no.

The patient may suffer from flu

Drink hot water, avoid cold eatables.
Drink lots of water and exercise regularly.
Drink salt and water.
Put hot water cloth on head and take crocin.
Drink salt and water, eat fruits.
true .
```

Conclusion:

Implementing an expert system using Prolog allows us to leverage its powerful logical reasoning capabilities to provide advice, solve problems, or make decisions in various domains. By encoding domain-specific knowledge in the form of facts and rules, we can build intelligent systems that mimic the decision-making abilities of human experts. Prolog's declarative nature and built-in inference engine make it a suitable choice for developing expert systems that require logical reasoning and knowledge representation.