Quiz11-Prolog

Due No due date **Points** 15 **Questions** 15
Available after Apr 29 at 1:20pm **Time Limit** 10 Minutes

Attempt History

	Attempt	Time	Score
LATEST	Attempt 1	1 minute	8 out of 15

Correct answers are hidden.

Score for this quiz: **8** out of 15
Submitted May 4 at 1:16pm
This attempt took 1 minute.

Question 1

1 / 1 pts

In the **PLP** paradigm, sometimes we must relax the separation of logic and control to gain programming flexibility and increase the user's domain, hence Prolog was introduced.

[✓ True]

False

Incorrect

Question 2

0 / 1 pts

In **PLP**, there is no negation via the "not" operator, i.e., we cannot write "X \= Y" (X does not equal Y).

True

[✓ False]

Question 3

1 / 1 pts

In Prolog, the first step of any “**goal**” deduction is always the ***resolution*** process.

[✓ True]

False

Incorrect

Question 4

0 / 1 pts

The ***cut*** (**!**) clause in Prolog mixes logic and control, hence Prolog is not **PLP** paradigm.

True

[✓ False]

Incorrect

Question 5

0 / 1 pts

The ***cut*** "!", “***fail***”, and “***asserted***” operators in Prolog are universally asserted.

[✓ True]

False

Incorrect

Question 6

0 / 1 pts

A non-terminating (upper case name) symbol parameter in any Prolog program *goal* clause is similar to an input/output parameter in Ada, to pass values and returns (implicitly some values) in the deduction process.

[✓ True]

False

Incorrect

Question 7

0 / 1 pts

The *top-down* depth-first deduction mechanism is much better than all other deduction mechanisms, since it gets to the first solution much faster than them.

[✓ True]

False

Question 8

1 / 1 pts

When a *cut* "!" operator is included in a Prolog rule, there is no way to try an alternative of such rule, even if it exists, in case we fail to prove the assertion of the right-hand side clauses (sub-goals) of such rule.

True [✓ False]

Question 9

1 / 1 pts

Yes, in Prolog we can write self-modifying code, where the programmer can modify the program’s database by adding/removing codes via the deduction process.

[✓ True] False

Question 10

1 / 1 pts

In Prolog, the order of facts, rules, and multiple subgoals in the right-hand-side of a rule are important.

[✓ True] False

Question 11

1 / 1 pts

Recursive rules in Prolog are always ***insecure*** and ***inefficient***, yet very powerful.

True [✓ False]

Question 12

1 / 1 pts

It might be more efficient if the facts are intermixed with the rules, in Prolog code (database).

[✓ True] False

Incorrect

Question 13

0 / 1 pts

Prolog is more amenable for concurrent processing than PLP languages.

[✓ True] False

Question 14

1 / 1 pts

The following Prolog clause is a *rule* (syntax), yet also a *fact* (semantics): **not(X):- fail.**

True [✓ False]

Incorrect

Question 15

0 / 1 pts

Prolog program **goal** must have at least one nonterminating symbol parameter (i.e., upper case symbols).

[✓ True] False

Quiz Score: **8** out of 15