AI Assignment

Question#1;

Let Suppose:-

at time t green light glow $\rightarrow gt$
at time t yellow light glow $\rightarrow yt$
at time t red light glow $\rightarrow rt$

- At any given moment, the traffic light is either green, yellow, and red:-

  $- \exists t\,(gt \vee yt \vee rt)$
  $- \; gt \vee yt \vee rt$

- The traffic light switches from green to yellow, yellow to red, red to green.

  $- \exists t\,((gt \wedge yt{+}1) \vee (yt \wedge rt{+}1) \vee (rt \wedge gt{+}1))$
  $- (gt \rightarrow yt{+}1) \vee (yt \rightarrow rt{+}1) \vee (rt \rightarrow gt{+}1)$

- The traffic light cannot remains in the same state for more than 3 consecutive cycles:

  $-(gt \wedge gt{+}1 \wedge gt{+}2) \vee (yt \wedge yt{+}1 \wedge yt{+}2) \vee$
  $(rt \wedge rt{+}1 \wedge rt{+}2)$

-

Question 2:-

$Con(x_1, x_2) \rightarrow x_1$ connects with x_2

$NC(x, cl) \rightarrow x$ has color cl

1. $\forall x \in N, \{x_1, x_2\} \in N, \{C_1, C_2\} \in C$

$- \{N(x_1) \wedge N(x_2) \wedge NC(x_1, c_1) \wedge NC(x_2, c_2) \wedge Con(x_1, x_2) \rightarrow c_1 \neq c_2\}$

$- Con(x_1, x_2) \rightarrow \neg (NC(x_1, c_1) \wedge NC(x_2, c_1))$

2. $(x_1 != x_2) \wedge NC(x_1, c_1) \wedge NC(x_2, c_1) \wedge (c_1 == \text{'yellow'}) \wedge$
$(z \in N: NC(z, c_1)) \rightarrow (z == x \vee z == y)$.

3. $\forall x_1 \in N \{NC(x_1, c_1 = R) \rightarrow NC(x_2, c_2 = G) \wedge steps(x_1, x_2) \leq 4\}$
$\exists x_2 \in N$

4. $\exists c_1 \in C, \{NC(x, c)\}$
$\forall x_1 \in N$

5.

Question 3:

Let's define statements:

$R(x) \rightarrow$ Read, x
$L(x) \rightarrow x$ is Literate
$D(x) \rightarrow x$ is Dolphin
$I(x) \rightarrow x$ is intelligent

- whoever can read is literate.

  - PL:→

    $R \rightarrow L$

    Read Literate

  - FOL:→

    $\forall x \{R(x) \rightarrow L(x)\}$

- Dolphins, unfortunately, are not literate.

  - PL:→

    $D \rightarrow \neg L$

    Dolphins

  - FOL

    $\forall x \{D(x) \longrightarrow \neg L(x)\}$

- Some dolphins are intelligents.

  - PL:→

    $D \land I$

    Dolphins Intelligents

  - FOL:→

    $\neg \forall x\, (D(x) \overset{\lor}{\cancel{\land}} \neg I(x))$ (or)

    $\exists x\, (D(x) \land I(x))$

- Some whos are ~~not~~ intelligent cannot read.

  - PL:→

    $\neg I \land \neg R$

  - FOL:→

    $\exists x\, (I(x) \land \neg R(x))$

    $\neg \forall x\, (I(x) \land R(x))$

- There exist a dolphin who is both intelligents and can read, but Every intelligent, if it can read, it must be that it not literate.

  - PL:→

    $(D \land I \land R) \land (D \land I \land R \land \neg R)$

  - FOL:→

    $\exists x\, (D(x) \land I(x) \land R(x)) \land$

    $\forall x\, (D(x) \land I(x) \land R(x) \rightarrow \neg L(x))$

→ Satisfiability:-

1. $\neg R \vee L$ $\{R \rightarrow L\}$
2. $\neg D \vee \neg L$ $(D \rightarrow \neg L)$
3. $D \wedge I$
4. $I \wedge \neg R$
5. $(D \wedge I \wedge R) \wedge (\neg I \vee \neg R \vee \neg D \vee \neg L)$

2 $\quad$ 3 $\quad$ 1

$\neg D \vee \neg L$; $D \wedge I = \{D, I\}$; $\neg R \vee L = \{R, L\}$

$D \wedge I \wedge \neg L$ $\{D, I, \neg L\}$ $\quad$ $\neg(D \wedge I \wedge R) \vee L$ $\{D, I, R, L\}$

4 $\quad$ 5 $\quad$ $I \wedge R$ (I, R)

$(I, \neg L, R)$

$L \wedge \neg L$

Φ Empty clause

Computation:-

Q.2 (a)

	Time Complexity	Memory Usage
initialize datalog()	$O(n+m)$ n — no. of stops m — no. of routes.	$O(n+m)$
add route data()	$O(m \times n)$ m – no. of routes. n – avg. stop/route	$O(m)$ m – no. of routes.
query direct routes()	$O(1)$	$O(k)$ k – direct no. of routes.

(b) Intermediate Step Comparison,

Brute force:-

↳ Iterate through each route
↳ Check if both stops exist in the route
↳ verify the order of stops.

FOL Approaches:-

↳ Create terms & Predicates
↳ Add facts to knowledge base
↳ Define relationship rules

Q.3) (a)

	Time Complexity	Space Complexity
forward_chaining	$O(n \times m^2)$	$O(n \times m) + p$
	n - no. of routes m - avg. stop/route.	p - no. of potential paths.
backward_chaining	$O(n \times m)$	$O(n \times m + G)$
		G - no. of goal-directed paths.

(b) forward Chaining :-

- Initialize knowledge base with route data.
- Generate Connected stops relationships
- Build paths forward from start_stop.
- Going apply via_stop Constraints.
- Return paths.

Backward chaining :-

- Same as above (forward Chaining)

but here we moves ~~start~~ end → ~~end~~ start rather than start → end.