Q1. Cities are located on a one-way road. Show how to complete the following program so as to check whether one can travel from one city to another.

oneWayRoad([lussac, gayac, figeac, trelissac, tourtoirac, dignac, fronsac, agonac, jumillac]).

travel(City1, City2) :-
    oneWayRoad(R),
    path(R, City1, City2).

Q2. Can the following pairs of predicates be unified (provide the resulting substitutions if yes).

a. \( p(X, f(X), Z) \) and \( p(g(Y), f(g(b)), Y) \)

b. \( p(X, f(X)) \) and \( p(f(Y), Y) \)

c. \( p(X, f(Z)) \) and \( p(f(Y), Y) \)

Q3. Consider the following axioms:

1. Every child loves Santa.
2. Everyone who loves Santa loves any reindeer.
3. Rudolph is a reindeer, and Rudolph has a red nose.
4. Anything which has a red nose is weird or is a clown.
5. No reindeer is a clown.
6. Scrooge does not love anything which is weird.
7. (Conclusion) Scrooge is not a child.

Represent these axioms in predicate calculus; convert each formula to clause form. (Notes: ‘has_a_red_nose’ can be a single predicate. Remember to negate the conclusion.) Prove the unsatisfiability of the set of clauses by resolution.
Q4. Provide a model in which \((\forall x) (P(x) \Rightarrow (\forall y) P(y))\) is true.

Q5. The following DCG recognizes an xml tag (we suppose that input is given as a list of ASCII codes):

\[
\text{tag}(S) \rightarrow [60], \text{str}(S), [62]. \quad \% \text{ 60 is the code for '<' and 62 for '>'}
\]
\[
\text{str}([X|S]) \rightarrow [X], \text{str}(S), \{X \lt\lt 60, X \lt\lt 62\}.
\]
\[
\text{str}([ ]) \rightarrow [ ].
\]

Write DCG predicate \textit{xml} that checks whether xml tags are well balanced. For instance, \textit{xml} should succeed on the string:

"<x1>I know that <h1>Prolog </h1>is logical</x1>"

but is should fail on

"<x1>I know that <h1>Prolog </x1>is logical</h1>"

(note: code for '/' is 47).

Q6. Provide the best generalization (lgg) for these two examples of the concept \textit{nice_food}:

\[
nice\text{-}food(X) :- \text{fruit}(X), \text{round}(X), \text{red}(X), \text{juicy}(X).
\]
\[
nice\text{-}food(X) :- \text{edible}(X), \text{yellow}(X), \text{sweet}(X), \text{has\_seeds}(X).
\]

using the background knowledge:

\[
edible(X) :- \text{fruit}(X).
\]
\[
juicy(X) :- \text{sweet}(X).
\]
\[
edible(X) :- \text{sweet}(X).
\]
Q1. Cities are located on a one-way road. Show how to complete the following program so as to check whether one can travel from one city to another.

```prolog
oneWayRoad([lussac, gayac, figeac, trelissac, tourtoirac, dignac, fronsac, agonac, jumillac]).

travel(City1, City2) :-
    oneWayRoad(R),
    path(R, City1, City2).

path([City1|R], City1, City2) :-
    !, % cut for efficiency
    member(City2, R).

path([_|R], City1, City2) :-
    path(R, City1, City2).
```

Q2. Can the following pairs of predicates be unified (provide the resulting substitutions if yes).

a. \( p(X, f(X), Z) \) and \( p(g(Y), f(g(b)), Y) \)

b. \( p(X, f(X)) \) and \( p(f(Y), Y) \)

c. \( p(X, f(Z)) \) and \( p(f(Y), Y) \)

a. Yes: \( X = g(b), Z = Y, Y = b \).

b. No. We would need \( Y = f(f(Y)) \), but recursive unification should fail (note: most Prolog implementation do not check for recursion for efficiency purposes).

c. Yes: \( X = f(f(Z)), Y = f(Z) \).

Q3. Consider the following axioms:

1. Every child loves Santa.
2. Everyone who loves Santa loves any reindeer.
3. Rudolph is a reindeer, and Rudolph has a red nose.
4. Anything which has a red nose is weird or is a clown.
5. No reindeer is a clown.
6. Scrooge does not love anything which is weird.
7. (Conclusion) Scrooge is not a child.

Represent these axioms in predicate calculus; convert each formula to clause form. (Notes: ‘has_a_red_nose’ can be a single predicate. Remember to negate the conclusion.) Prove the unsatisfiability of the set of clauses by resolution.

1. \[ (\forall x) \ (\text{child}(x) \supset \text{loves}(x, \text{santa})) \]
2. \[ (\forall x) \ (\text{loves}(x, \text{santa}) \supset (\forall y) \ (\text{reindeer}(y) \supset \text{loves}(x, y)))) \]
3. \[ \text{reindeer}(\text{Rudolph}) \land \neg \text{has_a_red_nose}(\text{Rudolph}) \]
Q4. Provide a model in which $(\forall x) (P(x) \Rightarrow (\forall y) P(y))$ is true.

Consider a model with a single element \{a\}.

Q5. The following DCG recognizes an XML tag (we suppose that input is given as a list of ASCII codes):

```
tag(S) --> [60], str(S), [62].   % 60 is the code for '<' and 62 for '>'
str([X|S]) --> [X], str(S), {X \== 60, X \== 62}.
str([ ]) --> [ ].
```

Write DCG predicate `xml` that checks whether XML tags are well balanced. For instance, `xml` should succeed on the string:

"<x1>I know that <h1>Prolog </h1>is logical</x1>"

but is should fail on

"<x1>I know that <h1>Prolog </x1>is logical</h1>"

(note: code for '/' is 47).

```
xm --> tag(T), text, tag([47|T]), {write('recognized: '), writef(T), nl}.
text --> str(_), xml, text.
text --> str(_).

?- xml('"<x1>I know that <h1>Prolog </h1>is logical</x1>"', []).
'recognized: h1
'recognized: x1
true .
```
?- xml('<xml>I know that <hl>Prolog </hl>is logical</hl>', []).  false.

Q6. Provide the best generalization (lgg) for these two examples of the concept nice_food:

nice_food(X) :- fruit(X), round(X), red(X), juicy(X).

nice_food(X) :- edible(X), yellow(X), sweet(X), has_seeds(X).

using the background knowledge:

edible(X) :- fruit(X).

juicy(X) :- sweet(X).

edible(X) :- sweet(X).

the lgg: nice_food(X) :- edible(X), juicy(X).