Name: Shah Farzan Al-Hafiz
ID: 10162590
CPSC449 – T01

Answer 1

1. The cat sat on a mat.
   sat(cat, mat).
2. Mary had a little lamb.
   had(mary, little_lamb).
3. The lamb's fleece was white as snow.
   fleece(lamb, white_as_snow).
4. If Mary went there, then so did the lamb.
   went(lamb, X) :- went(mary, X).
5. If a thing is both human and totally consistent, then the thing is dead.
   is(thing, dead) :- is(thing, human), is(thing, consistent).

Answer 2

1. scaredyCat(scooby_doo).
   Scooby Doo is a scaredy cat.
2. lessThan(1, 3).
   1 is less than 3.
3. likes(ice_cream, bill).
   ice cream likes Bill.
4. Has_red_hair(bill).
   Bill has red hair.
5. red(X) :- rose(X).
   If X is a rose, X is red.
6. even(zero).
   Zero is even.
7. even(suc(suc(N))) :- even(N).
   If N is even, then so is 1 + 1+ N.

Answer 3

```
odd(suc(zero)).
odd(suc(suc(X))) :-
    odd(X).
```

Answer 4

1. Reflexive means X is related to itself.
   ```
   ltReflexive(X, Y) :-
                   lt(X, X).
   ```
2. Symmetric means if X is related to Y, then Y is related to X.
   ```
   ltSymmetric(X, Y) :-
                    lt(X, Y), lt(Y, X).
   ```
3. Transitive means if X is related to Y, and Y is related to Z, then X is related to Z.
   ```
   ltTransitive(X, Z) :-
                    lt(X, Y), lt(Y, Z).
   ```
4. ltAntisymmetric(X, Y) :-
   ```
                           lt(X, Y), lt(Y, X), ltReflexive(X, Y).
   ```

Answer 5

1.
   ```
   app([], X, X).
   app([H|L1], X, [H|L2]) :-
         app(L1, X, L2)
   ```
2.
   ```
   reversed([],[]).
   reversed([H|T], L1) :-
               reversed(T, L2), app(L2, [H], L1).
   ```
3.
   ```
   palindromic(L) :-
                    reversed(L, L).
   ```
4.
   ```
   subset1(_,[]).
   subset1([H|L],[H|LS]) :-
              subset1(L,LS).
   subset(_,[]).
   subset([H|L],[H|LS]) :-
            subset1(L,LS).
   subset([_|L], LS) :-
            subset(L,LS).
   subset2(L, LS):-
   setof(X, subset(L, X), LS).
   ```
5.
   ```
   permutationList([], []).
   permutationList([X], [X]) :- !.
   permutationList([T|H], X) :-
              permutationList(H, H1), app(L1, L2, H1), app(L1, [T], X1), app(X1, L2, X).
   permutationList1(L, LS) :-
               setof(X, permutationList(L, X), LS).
   ```
6. Yes.

Answer 6

Bonus

```
gcd(0, N, N).
gcd(M, M, M).
gcd(M, N, D) :-  M > N, X is M - N, gcd(N, X, D).
gcd(M, N, D) :-  M < N, X is N - M, gcd(X, M, D)
```

```
1. divisible(A, B) :-
                 0 is A mod B.

   divisible(A, B) :-
                 B < A-1, divisible(B, Y+1).

   prime(2) :- !.
   prime(3) :- !.
   prime(A) :-
             A > 3, not(divisible(A, 2)).
```

Answer 7

```
1. changeLabel(_, _, X, X).
   changeLabel(X,Y, node(X, X1, X2),node(Y, Y1, Y2)) :-
               changeLabel(X,Y, X1, Y1), changeLabel(X, Y, X2, Y2).
```

2. Gave Up on it ▮▮ ....

Answer 8

Take the element X and transfer to the output sublist. If X and Y are not equal then the 2 intermediary elements has to have the same head. When there are no more Xs in the head of the first two sets you put X as the fourth argument. But if the element that is to be transferred is different than X, then transfer X.

Answer 9

```
   parents(X, Y) :-
             mom(X, Y).
   parents(X, Y) :-
             dad(X, Y).
   married(X, Y) :-
                married(Y, Z), Z = X, !.
```

```
 siblings(X, Y) :-
             parents(Z, X), parents(Z, Y), X \= Y.
 grandparents(X, Y) :-
                     parents(Z, Y), parents(X, Z).
 uncleOrAunt(X, Y) :-
                         parents(Z, Y), siblings(Z,X).
 cousins(X,Y) :-
             uncleOrAunt(Z,X), parents(Z,Y).
descendents(X, Y):-
                   parents(Y, X).
descendents(X, Y):-
                   parents(Z, X),descendents(Z, Y).
ancestors(X, Y) :-
              descendents(Y, X).
relative_of(X, Y) :-
                   ancestors(X, Y).
relative_of(X, Y) :-
                  ancestors(Y, X).
relative_of(X, Y) :-
                  (ancestors(Z, X), ancestors(Z, Y)), X \= Y.
extended_relative(X, Y) :-
                              married(M, F), relative_of(X, M), relative_of(Y, F)
wealthy_relative(X, Y):-
                             extended_relative(X, Y), wealthy(X).
```