```
/* Code to demonstrate decomposition (or aggregation) of
   algebraic formulae in Prolog.

   Michael E. Sparks, 6 Nov 2020

SAMPLE USAGE:

?- formula_check_tl(42*(-26)-5+1/(-1)).

Principal functor is +
A = '42* -26-5' and is odd
B = '1/ -1' and is odd
Y = '-1098' and is even
true.

?- formula_check(938+172*104/2+(-26)-5).

Principal functor is -
A = '938+172*104/2+ -26' and is even
B = '5' and is odd
Y = '9851' and is odd

Principal functor is +
A = '938+172*104/2' and is even
B = '-26' and is even
Y = '9856' and is even

Principal functor is +
A = '938' and is even
B = '172*104/2' and is even
Y = '9882' and is even

Principal functor is /
A = '172*104' and is even
B = '2' and is even
Y = '8944' and is even

Principal functor is *
A = '172' and is even
B = '104' and is even
Y = '17888' and is even
true.
*/

% perform top level analysis of overall formula
formula_check_tl(Formula) :-
    Formula =.. [Functor,A,B],
    display_formula_check(Functor,A,B).

% recursively analyze formula until all operands are atomic
formula_check(Formula) :-
    atomic(Formula), !.

formula_check(Formula) :-
    Formula =.. [Functor,A,B],
    display_formula_check(Functor,A,B),
    formula_check(A),
    formula_check(B).
```

```
display_formula_check(Functor,A,B) :-
    odd_or_even(A,A_type),
    odd_or_even(B,B_type),
    Formula =.. [Functor,A,B],
    Y is Formula, % recall what the meaning of is is!
    odd_or_even(Y,Y_type),
    nl, write("Principal functor is "), write(Functor),
    nl, write("A = '"), write(A), write("' and is "), write(A_type),
    nl, write("B = '"), write(B), write("' and is "), write(B_type),
    nl, write("Y = '"), write(Y), write("' and is "), write(Y_type),
    nl.

% Binary predicate using Prolog's if-then-else construct
% (';' implements disjunction, just as ',' implements conjunction)
odd_or_even(X,Y) :-
    (  X mod 2 =:= 0
    -> Y = "even"
    ;  Y = "odd"
    ).
```