Prolog Practice 2 - Solutions

For each question, include your **code & sample output**.

For the first 2 problems, assume there are facts in the database like the following:

```
parent(bob,alice).
sister(alice,steve).
brother(steve,alice).
grandparent(mike,steve).
```

Note that this is *not* a complete database. You will need to add additional facts to test the rules you write below.

1. Write a Prolog rules **`uncle/2`** and **`aunt/2`**.

```
uncle(A,B) :- brother(A,X) , parent(X,B).
aunt(A,B) :- sister(A,X) , parent(X,B).
```

2. Write a Prolog rule for **`cousin/2`** where the parameters are cousins if they share a grandparent. (For simplification, brothers & sisters can be considered cousins.)

```
cousin(A,B) :- grandparent(X,A), grandparent(X,B).
```

3. Write a Prolog program **`fib/2`** to compute the nth Fibonacci number. For example:

```
1 ?- fib(10,X).
X = 55.
```

```
fib(0,0) :- !.
fib(1,1) :- !.
fib(N,F) :-
      N>1,
      N1 is N-1,
      N2 is N-2,
      fib(N1,F1),
      fib(N2,F2),
      F is F1 + F2.
```

4. Write Prolog rules for **car/2** and **cdr/2** that act the same as car/cdr in Scheme. For example:

```
1 ?- car([a,b,c],a).
true.
2 ?- cdr([a,b,c],X).
X = [b, c].
```

```
car([H | _T], H).
cdr([_H | T], T).
```

5. Write a Prolog rule **double/2** where the 2nd argument is the 1st argument doubled. **Your rule should work for integers and lists.** For example:

```
double(3,6).
double([1,2,3],[2,4,6]).
```

```
double([],[]) :- !.
double([H | T], [H2 | T2]) :- !, double(H, H2), double(T,T2).
double(X, Y) :- number(X), !, Y is X * 2.
double(X, X).
```

6. Write a Prolog rule `member_of/2` that succeeds if e is a member of list and fails otherwise.

```
member_of(X, X) :- !.
member_of(X, [X | _T]) :- !.
member_of(X, [_H | T]) :- !, member_of(X, T).
```

7. Write a Prolog rule `get_odds/2` where the first parameter is a list and the 2nd parameter is a list of the odd numbers in the list.

```
is_odd(H) :- 1 is mod(H,2).
get_odds([],[]).
get_odds([H | T], [H | T1]) :- is_odd(H), !, get_odds(T,T1).
get_odds([_H | T], T1) :- get_odds(T,T1).
```

8. Write a Prolog rule `sum_all/2` where the first parameter is a list and the 2nd parameter is the sum of all elements of the list.

```
sum_all([],0) :- !.
sum_all([H|T], Sum) :- sum_all(T,TSum), Sum is TSum + H.
```

9. Write a Prolog rule `mult_all/2` where the first parameter is a list and the 2nd parameter is the product of all elements of the list.

```
mult_all([],0) :- !.
mult_all([H],H) :- !.
mult_all([H|T], Sum) :- mult_all(T,TSum), Sum is TSum * H.
```

10. Write a Prolog rule `square_list/2` where the first parameter is a list and the 2nd parameter is a list of the elements in the first list squared.

```
square_list([],[]) :- !.
square_list([H|T], [H2|T2]) :- H2 is H ** 2, square_list(T, T2).
```