Esercizio 1

Si scriva un predicato Prolog che data una lista ed un elemento El appartenente alla lista, restituisca in uscita l'elemento successivo ad El nella lista. Nel caso in cui El sia l'ultimo elemento il predicato dovrà fallire.
Esempio:

```
?- consec(3, [1,7,3,9,11],X).
   yes X=9
```

Soluzione 1:

```
consec(El, [El| [X|_]],X):-!.
consec(El, [_|Tail],X):- consec(El,Tail,X).
```

Esercizio 2

Avendo definito

```
pari(X) :- 0 is X mod 2.
```

definire il predicato split(+L,?P,?D) = se L `e una lista di interi, P `e la lista contenente tutti gli elementi pari di L e D tutti quelli dispari (nello stesso ordine in cui occorrono in L).
Esempio:

```
?-  split([1,2,3,7,8],P,D).
yes, D=[1,3,7], P=[2,8]
```

Soluzione 2

```
pari(X) :- 0 is X mod 2.
split([],[],[]).
split([X|R],[X|P1],D):-
   pari(X),!,
   split(R,P1,D).
split([X|R],P,[X|D1]):-split(R,P,D1).
```

Esercizio 3

Data una lista L1 e un numero intero N, scrivere un predicato Prolog domanda1(L1,N,L2) che restituisca in L2 la lista degli elementi di L1 che sono liste contenenti solo due valori interi positivi fra 1 e 9 la cui somma valga N.
Esempi:

```
?- domanda1([[3,1],5,[2,1,1],[3],[1,1,1],a, [2,2]],4,L2).
        yes, L2 = [[3,1], [2,2]]
```

Soluzione 3:

```
domanda1([],N,[]).
domanda1([[A,B]|R ], N, [[A,B]|S]):-
        N is A + B, !,
        domanda1( R,N,S ).
domanda1([_|R], N,S ):- domanda1( R,N,S ).
```

Esercizio 4

Si definisca un predicato in PROLOG chiamato averStud che applicato a un numero di matricola di uno studente Matr e a una lista di esami LE dia come risultato la media AV dei suoi voti. Ogni esame sia rappresentato da un termine della lista LE della forma esame(Matr,Esame,Voto). Si definisca prima la versione ricorsiva e poi quella ricorsiva-tail.

Esempio:

```
?-averStud(s1,[esame(s2,f1,30),
   esame(s1,f1,27),esame(s3,f1,25),
   esame(s1,f2,30)], AV).
   yes, AV = 28.5
```

Soluzione:

```
% versione ricorsiva
averStud(S,L,AV) :-
        totStud(S,L,N,T),
        N > 0,
        AV is T/N.
totStud(_,[],0,0) :- !.
totStud(S,[esame(S,_,V)|R],N,T) :- !,
        totStud(S,R,NN,TT),
        N is NN + 1,
        T is TT + V.
totStud(S,[_|R],N,T) :-
        totStud(S,R,N,T).

% versione tail-ricorsiva
averStud(S,L,AV) :-
        totStud(S,L,0,N,0,T),
        N > 0,
        AV is T/N.
totStud(_,[],N,N,T,T) :- !.
totStud(S,[esame(S,_,V)|R],NI,NO,TI,TO):- !,
        N is NI + 1, T is TI + V,
        totStud(S,R,N,NO,T,TO).
totStud(S,[_|R],NI,NO,TI,TO) :-
        totStud(S,R,NI,NO,TI,TO).
```