Esercizi su Prolog per Fondamenti di IA

2° Esercitazione

Esercizio 2.1

a) Si scriva un predicato che fornisca la lunghezza di una lista, intesa come numero di elementi, sia in versione ricorsiva che tail ricorsiva.

len(L,N) "N è la lunghezza (numero di elementi) di L" (realizzazione ricorsiva)
len1(L,N) "N è la lunghezza (numero di elementi) di L" (realizzazione iterativa)

Esempi:

```
?- len([a,b,c],N).
N = 3;
no

?- len([a,b,[c,d,e]],N).
N = 3;
no
```

b) Si scriva un predicato che mostri un elemento se questo appartiene ad una lista o, ricorsivamente, ad una lista che sia elemento della precedente.

depth_member(T,L) "T appartiene alla lista L o, ricorsivamente, ad una lista che appartiene a L"

Esempio:

```
?- depth_member(X,[a,b,[c,d,e]]).
X = a;
X = b;
X = [c,d,e];
X = c;
X = d;
X = e;
no
```

c) Si scriva un predicato che concateni due liste in una terza.

append(L1,L2,L) "L è la concatenazione delle liste L1 e L2"

Esempio:

```
?- append([a,b],[c],L).
L = [a,b,c];
no
```

d) Si scriva un predicato che inverta l'ordine degli elementi di una lista, sia in versione ricorsiva che tail ricorsiva.

reverse(L1,L2) "L2 è la lista inversa di L1" (realizzazione ricorsiva)
reverse1(L1,L2) "L2 è la lista inversa di L1" (realizzazione iterativa)

Esempi:
?- reverse([a,b,c],L).
L = [c,b,a];
no

?- reverse([a,b,[c,d,e]],L).
L = [[c,d,e],b,a];
no

Esercizio 2.2
a) Si scriva un predicato che fornisca la lista intersezione di due liste.

intersection(X,Y,Z) "Z è l'insieme intersezione X ∩ Y"

Esempio:
?- intersection([1,2,3], [2,3,4], L).
L = [2,3];
no

b) Si scriva un predicato che determini l'ultimo elemento di una lista.

last(L,X) "X è l'ultimo elemento della lista L"

Esempi:
?- last([a,b,c],N).
X = c;
no

?- last([a,b,[c,d,e]],X).
X = [c,d,e];
no

c) Si scriva un predicato che, date due liste L1 e L2, verifichi se L1 è una sottolista di L2 (sia come subset, sia come sottolista propria).

subset(L1,L2) "L2 contiene tutti gli elementi di L1 nello stesso ordine"
sublist(L1,L2) "L2 contiene L1 come parte di sé stessa"

Esempi:
?- subset([1,2],[1,3,2,4]).
yes

?- subset([1,2],[2,3,1,4]).

no

?- sublist([1,2],[3,4,1,2,5]).
yes

?- sublist([1,2],[3,4,1,5,2]).
no

d) Si scriva un predicato che verifichi se una lista è palindroma (ossia uguale alla sua inversa).

palin(L) "L è una lista palindroma"

Esempi:
?- palin([a,b,c]).
yes

?- palin([a,b,a]).
no

e) Si scriva un predicato che elimini da una lista tutti gli elementi ripetuti.

del_rip(L1,L2) "L2 è la lista L1 privata di tutte le ripetizioni"

Esempio:
?- del_rip([a,b,b,c,a,d],L).
L = [a,b,c,d];
no

f) Si scriva un predicato che, dati un termine T e una lista L, conti le occorrenze di T in L.

conta(T,L,N) "N è il numero di occorrenze del termine T nella lista L"

Esempio:
?- conta(a,[b,a,a,b,c,a],N).
N = 3;
no

g) Si scriva un predicato che, data una lista ordinata di interi, inserisca in modo ordinato un altro numero intero, mantenendo la lista risultato ordinata.

insert(N,L,X) "X è la lista ordinata L in cui viene inserito in modo ordinato l'intero N"

Esempio:
?- insert(3,[1,2,3,4,5],X).
X = [1,2,3,3,4,5];
no

Esercizio 2.3
L'algoritmo di Euclide per il calcolo del massimo comun divisore è dato dalla seguente definizione:

$$mcd(A,B) = \begin{cases} A & se \quad A = B \\ mcd(A, B-A) & se \quad A < B \\ mcd(A-B, B) & se \quad A > B \end{cases}$$

Si consideri il seguente programma Prolog, che è l'implementazione dell'algoritmo di Euclide

```
mcd(A,A,A):-!.
mcd(A,B,M):- A<B,!, C is B-A, mcd(A,C,M).
mcd(A,B,M):- I is A-B, mcd(I,B,M).
```

Sebbene sembri corretto, l'algoritmo va in loop con il goal

?- mcd(2,2,1).

Si corregga il programma.

Esercizio 2.4
Sia i Jedi che i Sith sono entrambi eccellenti combattenti con le spada laser, tuttavia vi sono membri di entrambi gli schieramenti che abbandonano l'uso della spada singola per specializzarsi in uno stile di combattimento più esotico. Principalmente ci sono due specializzazioni oltre alla spada singola: il combattimento con due spade e con una spada a doppia lama.
Si scrivano i seguenti predicati:

doppiaspada(X) "X è specializzato nella coppia di spade"

doppialama(X) "X è specializzato nella spada a doppia lama"

Sapendo che:

a) Un individuo è specializzato nella doppia spada se rispetta i seguenti requisiti:
- non è specializzato nella spada singola
- è Jedi o Sith
- possiede due spade

b) Un individuo è specializzato nella spada a doppia lama se rispetta i seguenti requisit:
- non è specializzato nella spada singola
- è Sith o Jedi
- non è specializzato nella doppia spada

Avendo a disposizione i seguenti predicati:

spada(X) "X è specializzato nella spada singola"

jedi(X) "X è Jedi"

sith(X) "X è Sith"

haduespade(X) "X possiede due spade"

Sono dati i seguenti fatti:
sith(sidious).
sith(maul).
sith(exarkun).
sith(vader).
jedi(yoda).
jedi(kenobi).
jedi(macewindu).
jedi(ayalasecura).
spada(sidious).
spada(yoda).
spada(macewindu).
spada(kenobi).
spada(vader).
haduespade(ayalasecura).
haduespade(exarkun).

NB: si usi il **not** per la realizzazione dei requisiti negati (in SICStus si precede il predicato da negare con il simbolo \+).