الغاية من الجلسة: المتابعة في اللوائح.

مثال1:

المعلن الآتي append (وهو جاهز في البرولوغ أي أنه يمكنك أن تستخدمه دون أن تكتب قاعدة المعرفة له) يقوم بدمج لائحتين في لائحة جديدة حيث أن الوسيط الأول هو اللائحة الأولى والوسيط الثاني هو اللائحة الثانية أما الوسيط الثالث هو لائحة الخرج.

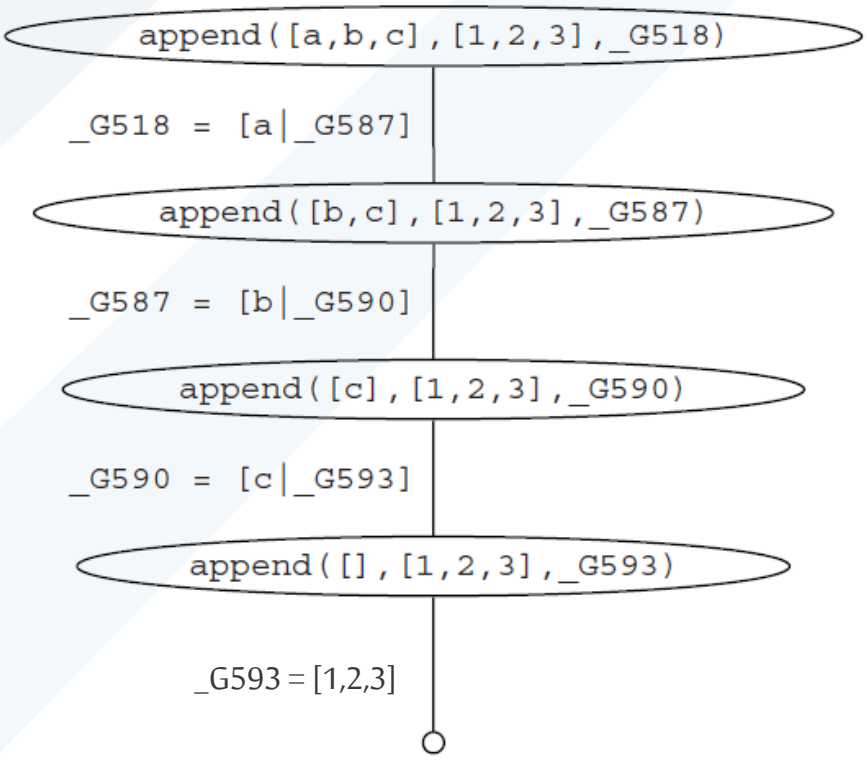
قاعدة المعرفة الخاصة بهذا المعلن هي:

```
append([],L,L).
append([H|T],L2,[H|L3]) :- append(T,L2,L3).
```

Examples:

```
?- append([a,b,c],[1,2,3],[a,b,c,1,2,3]).
  true.
?- append([a,b,c],[1,2,3],L3).
  L3 = [a,b,c,1,2,3].
```

الشكل الآتي يمثل الـ proof search الذي يوضح لك كيفية سير هذا المعلن:



مثال2 :

المعلن الآتي combine يدمج لائحتين في لائحة جديدة لكن بشكل مختلف عن السابق بحيث أنه يأخذ عنصر من اللائحة الأولى مع المقابل له في اللائحة الثانية:

```
combine([],[],[]).
combine([H1 | T1], [H2 | T2], [H1, H2 | T3]) :- combine(T1,T2, T3).
```

مثال عن المعلن السابق:

```
?- combine([1,4,3],[a,r,t],V).
V = [1, a, 4, r, 3, t].
```

مثال3:

المعلن الآتي odd يضع الأعداد الفردية الموجودة في لائحة أولى في لائحة ثانية:

```
odd([],[]).
odd([X|L],[X|Z]) :- 1 is mod(X,2),odd(L,Z).
odd([X|L],Z) :- 0 is mod(X,2), odd(L,Z).
```

في القاعدة الأولى نقصد أنه في حال كان باقي قسمة الرأس على 2 هو 1 بالتالي الرأس هو فردي لذلك ضعه في لائحة الخرج.

في القاعدة الثانية نقصد أنه في حال كان باقي قسمة الرأس على 2 هو 0 بالتالي الرأس هو زوجي لذلك **لا** تضعه في لائحة الخرج.

تمرين مع حل:

اكتب قاعدة المعرفة التي تعبر عن المعلن الآتي والذي يحذف الوسيط الأول من اللائحة التي هي الوسيط الثاني ويضع الخرج في لائحة جديدة وهي الوسيط الثالث:

```
?-deleteAll(a,[b, a, c, f, a],Z).
Z = [b, c, f].
```

الحل:

```
delete.([],[],_)
delete(A,[A|L],Z) :- delete(A,L,Z).
delete(A,[B|L],[B|M]) :- A =\=B, delete(A,L,M).
```

ملاحظة:

نقصد بالـ =\= بعدم المساواة أي A =\= B أي هل A لا يساوي B؟ أما عندم نكتب A =:= B نقصد بها هل A يساوي B؟

أي A =\= B تأخذ القيمة true عندما لا يتساوى A و B، أما A =:= B تأخذ القيمة true عندما يتساوى A و B.

تمرين للحل:

اكتب قاعدة المعرفة للمعلن removeDuplicates والذي يقوم بحذف التكرارات من لائحة، مثال:

```
?-removeDuplicates([a,b,b,c,a],R).
R = [b,c,a].
```

مساعدة في الحل: نختبر في كل مرة إذا كان الرأس ينتمي للذيل أم لا ، فإذا كان الرأس ينتمي للذيل بالتالي هو ما زال موجوداً في اللائحة بالتالي لا تضعه في لائحة الخرج R ، أما إذا كان الرأس لا ينتمي للذيل عندئذ ضعه في لائحة الخرج R.

مثال على معلن rev يقوم بعكس لائحة:

```
rev(L,R) :- accRev(L,[],R).
accRev([H|T],A,R) :- accRev(T,[H|A],R).
accRev([],A,A).
```

مثلاً:

```
?- rev([1,2,3,4,5],Y).
Y = [5, 4, 3, 2, 1].
```

الفكرة هو أنه نحن بحاجة من أجل أن نكتب قاعدة المعرفة وسيطاً إضافياً وهو المُراكم الذي سوف نضع القيم فيه، ولكن التابع المطلوب منا rev يمتلك فقط وسيطين L,R وبالتالي القاعدة الأولى هي فقط من أجل التحويل من الوسيطين إلى 3 وسطاء بحيث أن الوسيطين الأول والثالث في accRev هما نفسهما L,R بينما الوسيط الثاني فهو المراكم الذي نريد تعبئة العناصر به وتكون قيمته البدائية هي [].
القاعدة الثانية هي من أجل تعبة العناصر في المراكم، أما القاعدة الثالثة هي فقط من أجل التوقف.

تدريب مع حل:

اكتب معلناً avg يحسب معدل عناصر لائحة حيث الوسيط الأول هو اللائحة والوسيط الثاني هو الخرج الذي هو المعدل Average.
أي مثلاً:

```
?- avg([1,4,3,5,2],Avg).
Avg = 3.
```

الحل:

```
avg(List,Avg):-List=[H|T],avg1(List,Avg,0,0).
avg1([],Avg,Sum,I):-Avg is Sum/I.
avg1([H|T],Avg,Sum,I):-Sum1 is Sum+H,I1 is I+1,avg1(T,Avg,Sum1,I1).
```

لقد احتجنا وسيطين إضافيين وهما الأول من أجل تخزين المجموع والذي يجب أن يبدأ بالقيمة 0 أما الثاني فهو عداد من أجل حساب عدد عناصر اللائحة (من أجل قسمة المجموع عليه في النهاية) وأيضاً العداد يجب أن يبدأ بالقيمة 0.
ومن أجل كل مرة (أي من أجل كل رأس) نقوم بإضافة الرأس إلى المجموع وأيضاً زيادة العداد واحد.
ملاحظة: عملية الإسناد في البرولوغ هي is أي X is 3+4 يضع القيمة 7 في X.

تمارين للحل:

اكتب معلناً يقوم بإضافة عنصر إلى بداية لائحة، مثال:

```
?- add(5,[1,2,3] ,R).
R = [5, 1, 2, 3].

?- add(4,[],R).
R = [4].
```

اكتب معلناً يبحث عن عنصر ضمن لائحة ويعيد لنا مكان وجود العنصر لأول مرة في اللائحة، مثلاً:

```
?- search(4,[5,7,4,2,8,4,9],R).
R = 3.
```

عدّل المعلن السابق من أجل أن يعيد مواقع وجود العنصر في اللائحة، ويضع هذه المواقع ضمن لائحة جديدة، مثلاً:

```
?- newSearch(4,[5,7,4,2,8,4,9,4],R).
R = [3,6,8].
```