Lists

A list is an ordered sequence of objects and lists. In Prolog, a list is written as its elements separated by commas and enclosed in brackets. For example:

`[]`	This is an empty list.
`[peter, 4, abc, X]`	This is a list containing four elements. There is no restrictions on the number of variables and constants in the list.
`[a, b, [c, d], e]`	This list contains four elements. The third element is a list. This example shows that a list can also be an element of another list.

Head and tail

The head of a list is the first element in the list. The tail of a list is the list that remains after the first element is removed. For example:

List	Head	Tail
`[]`	undefined	undefined
`[a]`	`a`	`[]`
`[a, b]`	`a`	`[b]`
`[[a, b], c]`	`[a, b]`	`[c]`
`[[a, b], [c, d]]`	`[a, b]`	`[[c, d]]`

A list with `X` as the head and `Y` as the tail can be expressed as `[X | Y]`. See the following examples.

Examples

Assume that the following program is consulted:

Program 5: A program about lists

```
head([H | T], H).                       /* R1 */
tail([H | T], T).                       /* R2 */
list([H | T], H, T).                    /* R3 */

max([H], H).                            /* R4 */
max([H | T], H) :- max(T, X), H >= X.   /* R5 */
max([H | T], X) :- max(T, X), H < X.    /* R6 */
```

Finding heads and tails

The following queries make use of facts R1 through R3 and are quite obvious.

Query	Overall output

`?- head([], X).`	**`no`**
`?- list([], X, Y).`	**`no`**
`?- head([a, b, c, d], X).`	**`X = a`**
`?- tail([a], X).`	**`X = []`**
`?- list([a, b], X, Y).`	**`X = a`** **`Y = [b]`**

Finding maximum value

Note: In each step of the process of deduction, there may be more than one subgoals to be processed. In this case, only the first subgoal is considered. If there is a rule that can be applied to it, it is replaced by those subgoals found in the applied rule. If it satisfies a fact, it is removed. If no rules or facts can be applied, we have to backtrack.

Sample query:

```
?- max([3, 5, 2, 4], A).
```

Level	Results of deduction	Rules or facts to be applied	Bindings of variables
Original query	`max([3, 5, 2, 4], A).`	R5	`A = 3`
1	`max([5, 2, 4], X), 3 >= X.`	R5	`X = 5`
2	`max([2, 4], X), 5 >= X, 3 >= 5.`	R5	`X = 2`
3	`max([4], X), 2 >= X, 5 >= 2, 3 >= 5.`	R4	`X = 4`
4	`2 >= 4, 5 >= 2, 3 >= 5.`	Failed! Backtracks.	
3	`max([4], X), 2 >= X, 5 >= 2, 3 >= 5.`	R5	`X = 4`
4	`max([], X), 2 >= 4, 5 >= 2, 3 >= 5.`	Failed! Backtracks.	
3	`max([4], X), 2 >= X, 5 >= 2, 3 >= 5.`	R6	-
4	`max([], X), 4 < X, 2 >= 4, 5 >= 2, 3 >= 5.`	Failed! Backtracks.	
3	`max([4], X), 2 >= X, 5 >= 2, 3 >= 5.`	Failed! Backtracks.	
2	`max([2, 4], X), 5 >= X, 3 >= 5.`	R6	-
3	`max([4], X), 2 < X, 5 >= X, 3 >= 5.`	R4	`X = 4`
4	`2 < 4, 5 >= 4, 3 >= 5.`	Satisfied.	
5	`5 >= 4, 3 >= 5.`	Satisfied.	
6	`3 >= 5.`	Failed! Backtracks.	
5	`5 >= 4, 3 >= 5.`	Failed! Backtracks.	
4	`2 < 4, 5 >= 4, 3 >= 5.`	Failed! Backtracks.	
3	`max([4], X), 2 < X, 5 >= X, 3 >= 5.`	R6	-
4	`max([], X), 4 < X, 2 < X, 5 >= X, 3 >= 5.`	Failed! Backtracks.	
3	`max([4], X), 2 < X, 5 >= X, 3 >= 5.`	Failed! Backtracks.	

2	`max([2, 4], X), 5 >= X, 3 >= 5.`	Failed! Backtracks.	
1	`max([5, 2, 4], X), 3 >= X.`	R6	-
2	`max([2, 4], X), 5 < X, 3 >= X.`	R5	`X = 2`
3	`max([4], X), 2 >= X, 5 < 2, 3 >= 2.`	R4	`X = 4`
4	`2 >= 4, 5 < 2, 3 >= 2.`	Failed! Backtracks.	
3	`max([4], X), 2 >= X, 5 < 2, 3 >= 2.`	R5	`X = 4`
4	`max([], X), 4 >= X, 2 >= 4, 5 < 2, 3 >= 2.`	Failed! Backtracks.	
3	`max([4], X), 2 >= X, 5 < 2, 3 >= 2.`	R6	-
4	`max([], X), 4 < X, 2 >= X, 5 < 2, 3 >= 2.`	Failed! Backtracks.	
3	`max([4], X), 2 >= X, 5 < 2, 3 >= 2.`	Failed! Backtracks.	
2	`max([2, 4], X), 5 < X, 3 >= X.`	R6	-
3	`max([4], X), 2 < X, 5 < X, 3 >= X.`	R4	`X = 4`
4	`2 < 4, 5 < 4, 3 >= 4.`	Satisfied.	
5	`5 < 4, 3 >= 4.`	Failed! Backtracks.	
4	`2 < 4, 5 < 4, 3 >= 4.`	Failed! Backtracks.	
3	`max([4], X), 2 < X, 5 < X, 3 >= X.`	R5	`X = 4`
4	`max([], X), 2 < 4, 5 < 4, 3 >= 4.`	Failed! Backtracks.	
3	`max([4], X), 2 < X, 5 < X, 3 >= X.`	R6	-
4	`max([], X), 4 < X, 2 < X, 5 < X, 3 >= X.`	Failed! Backtracks.	
3	`max([4], X), 2 < X, 5 < X, 3 >= X.`	Failed! Backtracks.	
2	`max([2, 4], X), 5 < X, 3 >= X.`	Failed! Backtracks.	
1	`max([5, 2, 4], X), 3 >= X.`	Failed! Backtracks.	
Original goal	`max([3, 5, 2, 4], A).`	R6	-
1	`max([5, 2, 4], A), 3 < A.`	R5	`A = 5`
2	`max([2, 4], X), 5 >= X, 3 < 5.`	R5	`X = 2`
3	`max([4], X), 2 >= X, 5 >= 2, 3 < 5.`	R4	`X = 4`
4	`2 >= 4, 5 >= 2, 3 < 5.`	Failed! Backtracks.	
3	`max([4], X), 2 >= X, 5 >= 2, 3 < 5.`	R5	`X = 4`
4	`max([], X), 4 >= X, 2 >= 4, 5 >= 2, 3 < 5.`	Failed! Backtracks.	
3	`max([4], X), 2 >= X, 5 >= 2, 3 < 5.`	R6	-
4	`max([], X), 4 < X, 2 >= X, 5 >= 2, 3 < 5.`	Failed! Backtracks.	
3	`max([4], X), 2 >= X, 5 >= 2, 3 < 5.`	Failed! Backtracks.	
2	`max([2, 4], X), 5 >= X, 3 < 5.`	R6	-
3	`max([4], X), 2 < X, 5 >= X, 3 < 5.`	R4	`X = 4`
4	`2 < 4, 5 >= 2, 3 < 5.`	Satisfied.	
5	`5 >= 2, 3 < 5.`	Satisfied.	
6	`3 < 5.`	Satisfied!	

		Output: `A = 5` Then backtracks.	
5	`5 >= 2, 3 < 5.`	Failed! Backtracks.	
4	`2 < 4, 5 >= 2, 3 < 5.`	Failed! Backtracks.	
3	`max([4], X), 2 < X, 5 >= 2, 3 < 5.`	R5	`X = 4`
4	`max([], X), X >= 4, 2 < 4, 5 >= 2, 3 < 5.`	Failed! Backtracks.	
3	`max([4], X), 2 < X, 5 >= 2, 3 < 5.`	R6	-
4	`max([], X), 4 < X, 2 < X, 5 >= 2, 3 < 5.`	Failed! Backtracks.	
3	`max([4], X), 2 < X, 5 >= 2, 3 < 5.`	Failed! Backtracks.	
2	`max([2, 4], X), 5 >= X, 3 < 5.`	Failed! Backtracks.	
1	`max([5, 2, 4], A), 3 < A.`	R6	-
2	`max([2, 4], A), 5 < A, 3 < A.`	R5	`A = 2`
3	`max([4], X), 2 >= X, 5 < 2, 3 < 2.`	R4	`X = 4`
4	`2 >= 4, 5 < 2, 3 < 2.`	Failed! Backtracks.	
3	`max([4], X), 2 >= X, 5 < 2, 3 < 2.`	R5	`X = 4`
4	`max([], X), 4 >= X, 2 >= 4, 5 < 2, 3 < 2.`	Failed! Backtracks.	
3	`max([4], X), 2 >= X, 5 < 2, 3 < 2.`	R6	-
4	`max([], X), 4 < X, 2 >= X, 5 < 2, 3 < 2.`	Failed! Backtracks.	
3	`max([4], X), 2 >= X, 5 < 2, 3 < 2.`	Failed! Backtracks.	
2	`max([2, 4], A), 5 < A, 3 < A.`	R6	-
3	`max([4], A), 2 < A, 5 < A, 3 < A.`	R4	`A = 4`
4	`2 < 4, 5 < 4, 3 < 4.`	Succeeded.	
5	`5 < 4, 3 < 4.`	Failed! Backtracks.	
4	`2 < 4, 5 < 4, 3 < 4.`	Failed! Backtracks.	
3	`max([4], A), 2 < A, 5 < A, 3 < A.`	R5	`A = 4`
4	`max([], X), 4 >= X, 2 < 4, 5 < 4, 3 < 4.`	Failed! Backtracks.	
3	`max([4], A), 2 < A, 5 < A, 3 < A.`	R6	-
4	`max([], A), 4 < A, 2 < A, 5 < A, 3 < A.`	Failed! Backtracks.	
3	`max([4], A), 2 < A, 5 < A, 3 < A.`	Failed! Backtracks.	
2	`max([2, 4], A), 5 < A, 3 < A.`	Failed! Backtracks.	
1	`max([5, 2, 4], A), 3 < A.`	Failed! Backtracks.	
Original goal	`max([3, 5, 2, 4], A).`	Failed! There are no more rules or facts that can be applied.	
Overall output	`A = 5`		