

## Solutions to last week's exercises: I

- A program for filtering out elements greater than 20 in a list of numbers:

```
filter([], []).  
filter([H|T], [H|L]) :-  
    H>20,  
    filter(T, L).  
filter([H|T], L) :-  
    H=<20,  
    filter(T, L).
```

## Solutions to last week's exercises:II

- A program for removing all instances of an element from a list:

```
remove(Element, [], []).
remove(Element, [H|T], List):-
  Element = H,
  remove(Element, T, List).
remove(Element, [H|T], [H|List]):-
  Element / == H,
  remove(Element, T, List).
```

## More list exercises

- Write a program to replace all instances of `Element1` in a list with `Element2` (this should be easy if you understand the code for `remove`).
- Write a program to reverse the elements of a list. (Solution on next slide).
- Write a program to count the number of elements in a list.
- Write a program that swaps first and second, third and fourth elements of a list and so on.

## Reversing a list

```
reverse([], []).  
reverse([H|T], L2) :-  
    reverse(T, NT), append(NT, [H], L2).
```

Is there a better way of writing this program ?

## The CUT predicate

A program to implement: "Q is true iff P isn't"

```
Q :- P, !, fail.
```

```
Q.
```

- The `fail` predicate always fails.
- The predicate "cut" (written "!") tells Prolog not to pass back through this point when it is looking for alternative solutions. Thus, the "!" acts as a marker, back beyond which Prolog will not go. The cut predicate itself always evaluates to true. (More on cut next week).