Write a predicate member/2 that takes a list as its first argument and an element as its second element. This predicate is to return true if the element appears in the list.

\[
\text{member([E|\_],E).
member([\_|T],E) :- member(T,E).}
\]
Write a program that prompts a user for a list, then reads the list, reverses the elements of the list and then prints out the reversed list to the terminal. It then returns to prompting the user for a new list, etc.

```
interact:-
    nl, write('gimme a list> '), read(X), reverse(X,Y), write('this is the reverse: '), write(Y), nl, interact.
```
Write a program that takes simple English statements and translates them into German. The sentences are given as lists of words.

\[
\begin{align*}
\text{% the dictionary} \\
\text{lookup(logic, logik).} \\
\text{lookup(is, macht).} \\
\text{lookup(fun, spass).} \\
\end{align*}
\]

\[
\begin{align*}
\text{% the translation procedure} \\
\text{translate([], []).} \\
\text{translate([[Word|Sentence]], [German]):=} \\
\quad \text{lookup(Word, GWord),} \\
\quad \text{translate(Sentence, GSentence),} \\
\quad \text{German=[[GWord|GSentence]].} \\
\end{align*}
\]
Prolog Final Remarks

- Prolog has no explicit sequence control, the flow of control is driven by the pattern matching of the heads of the rules against the current (sub)goal statements.
- This has an effect on how we program - rather than explicit ‘how to’ statements we axiomatize the solution we are looking for, e.g.,
  - The length of an empty list is 0
  - The length of the overall list is the length of the rest of the list plus 1.
  - …rather than defining explicit iterations over record structures.
Assignment

- Assignment #12 – see website