A simple D-type flip-flop is represented by the Prolog predicate `dff` whose definition is as follows:

\[
\text{dff}(D, 0, Q, Q).
\]
\[
\text{dff}(D, 1, Q, D).
\]

The first argument is the input to the flip-flop, the second is the clock with 0 representing a falling edge and 1 representing a rising edge. The third and fourth arguments are the previous and next states of the flip-flop. As can be seen the state of the flip-flop changes on a rising edge of the clock.

A clocked circuit consists of three d-type flip-flops with inputs and states \((D_1, Q_1)\), \((D_2, Q_2)\) and \((D_3, Q_3)\). They are wired in such a way that

\[
D_1 = (Q_1 \land Q_2) \lor (\overline{Q_1} \land \overline{Q_2})
\]
\[
D_2 = (\overline{Q_1} \land Q_3) \lor (Q_2 \land \overline{Q_3})
\]
\[
D_3 = (Q_1 \land Q_3) \lor (\overline{Q_2} \land \overline{Q_3})
\]

(a) Using \(s(Q_1, Q_2, Q_3)\) to represent the state of the circuit, define a predicate that will compute the state after the next rising edge of the clock. You may find it helpful to define predicates to represent \textit{and}, \textit{or} and \textit{not} gates.

(b) Define a predicate `testcc(N, s(Q1,Q2,Q3), List)` that will compute the list of states \(\text{List}\) through which the circuit passes from the given initial state \(s(Q1,Q2,Q3)\) as a result of a sequence of \(N\) rising edges of the clock.