Question 1
Follow the method viewed in course (cf. slides): 1 init state at the beginning, one state per `await` statement.

Question 2
There are many solutions. This one uses a delayed switch: conversely to the classical one (cf. slides) that correspond to a time interval `[on, off]` (on included, off excluded), `dswitch` corresponds to an interval `(on, off)` (on excluded, off excluded).

```plaintext
node dswitch(on, off: bool) returns (state: bool);
let
    state = false -> pre (if state then not off else on);
end

node clics(c, t: bool) returns (S, D: bool);
var w1, w2: bool;
let
    assert #(c, t);
    w1 = dswitch(c, S or D);
    w2 = dswitch(w1 and t and not c, S or D);
    D = w1 and c;
    S = w2 and t;
end
```
**Question 3**

Structural constraints:

\[
\begin{align*}
a &= 1; \\
k &= 1; \\
a + j &= b + k; \\
b &= c + d; \\
c + d &= f + e; \\
e + f &= g + h; \\
g + h &= j; \\
\end{align*}
\]

Objective:

\[
\text{max: } 15 \cdot c + 10 \cdot d + 8 \cdot e + 15 \cdot f + 20 \cdot g + 12 \cdot h;
\]

**Question 4**

The loop is executed at most 10 times (in fact exactly 10, but it does not matter for WCET). New constraint: \( b \leq 10 \) or equivalently \( j \leq 10 \).

First estimation: 500, corresponding to take 10 times the path c.f.g

**Question 5**

During each iteration, it is impossible to take the sequence: c.f.j, because: the iteration starts with \( y = x[i] \) (b), then (c) \( z = 0 \), then \( y += z \) (f) and since \( z = 0 \) \( y \) does not change (\( y = x[i] \)), then \( y > x[i] \) is false and (g) is impossible.

For each iteration \( t \), we have \( c_t + f_t + g_t \leq 2 \), since there are 10 iterations: \( c + f + g \leq 20 \).

With this constraint, the worst path at each iteration is d.f.g (45) thus the WCET is \( 10 \cdot 45 = 450 \).
Question 6

The formula has no implicant of length 1 (i.e. none of $x, \bar{x}, y, \bar{y}, z, \bar{z}$ are implicant).

It has 2 implicants of length 2, that are thus prime :

(A) $\bar{x}.y$
(B) $y.\bar{z}$

And finally 5 implicants of length 3 :

$\bar{x}.y.z$, not prime since it implies (A)
$\bar{x}.y.\bar{z}$ not prime since it implies (A)
$x.y.\bar{z}$ not prime, not prime since it implies (B)
$x.\bar{y}.z$ prime

Question 7

Whenever the right branch is 1, we can deduce that the root variable $x$ implies the whole formula, thus $x$ is a (minimal) implicant :

$$I \left( \begin{array}{cc} x & 1 \\ 0 & 1 \end{array} \right) = x$$

Same reasoning when the left branch is 1 : $\bar{x}$ is a (minimal) implicant :

$$I \left( \begin{array}{cc} x & 0 \\ 1 & 1 \end{array} \right) = \bar{x}$$

When the left branch is 0, it means that $x$ MUST TRUE to satisfy the formula, i.e. $x$ necessarily appears in any implicant. To obtain a “minimal” implicant then : compute a “minimal” implicant for the right branch ($f_1$), and concatenates $x$ to the result :

$$I \left( \begin{array}{cc} x & 0 \\ f_0 & 1 \end{array} \right) = x \cdot I(f_1)$$

Similarly, when the left branch is 0, $\bar{x}$ is necessary to satisfy the formula, and thus it must appear in any implicant :

$$I \left( \begin{array}{cc} x & 1 \\ f_0 & 0 \end{array} \right) = \bar{x} \cdot I(f_0)$$

Finally, when none of the branches is a constant, we have to find a “strategy” that hopefully gives a “small” implicant. A reasonable strategy consist in searching, if it exists, an implicant where the variable $x$ does not appear (directly or negated). For that, we first build the consensus formula :

$$g = f_0 \cdot f_1$$

(Note : cf. course, $g = \forall x \ f$)
If $g$ is NOT IDENTICALLY 0, it means that any implicant of $g$ is also an implicant of $f$, and, since it does not contain the variable $x$, we can expect it to be small (with no guaranty!):

$$\left(\begin{array}{c}
x \\
\bigwedge \\
f_0 \\
f_1
\end{array}\right) = \mathcal{I}(g) \text{ if } g = f_0 \cdot f_1 \neq 0$$

If $g$ is IDENTICALLY 0, it means that $x$ or $\bar{x}$ is necessary in any implicant; in this case one can choose arbitrarily to compute an implicant with $x$ or with $\bar{x}$, for instance:

$$\left(\begin{array}{c}
x \\
\bigwedge \\
f_0 \\
f_1
\end{array}\right) = x \cdot \mathcal{I}(f_1) \text{ if } g = f_0 \cdot f_1 = 0$$

or:

$$\left(\begin{array}{c}
x \\
\bigwedge \\
f_0 \\
f_1
\end{array}\right) = \bar{x} \cdot \mathcal{I}(f_0) \text{ if } g = f_0 \cdot f_1 = 0$$

NOTE: with the previous definition, $\mathcal{I}$ is strictly linear w.r.t. the size of the operand, but has no guaranty on the quality (the minimality) of the result. We can have a better result, but with a much costly algorithm, (not only non-linear, but non polynomial!). The idea is simple: when there are several choices, computes all the cases, and keep the “best one”. Suppose that we have a procedure shortest that takes a list of monomial and returns one of them with a minimal length:

$$\left(\begin{array}{c}
x \\
\bigwedge \\
f_0 \\
f_1
\end{array}\right) = \text{shortest} (\mathcal{I}(g), x \cdot \mathcal{I}(f_1), \bar{x} \cdot \mathcal{I}(f_0)) \text{ if } g = f_0 \cdot f_1 \neq 0$$

$$\left(\begin{array}{c}
x \\
\bigwedge \\
f_0 \\
f_1
\end{array}\right) = \text{shortest} (x \cdot \mathcal{I}(f_1), \bar{x} \cdot \mathcal{I}(f_0)) \text{ if } g = f_0 \cdot f_1 = 0$$