## Some clues how to find invariants:

According to derived While rule we get 3 verification conditions:

$$\frac{\vdash P \Rightarrow R \qquad\qquad \vdash \{R \wedge S\}\ C\ \{R\} \qquad\qquad \vdash R \wedge \neg S \Rightarrow Q}{\vdash \{P\}\ \mathbf{WHILE}\ S\ \mathbf{DO}\ C\ \{Q\}}$$

(where *P*- precondition, *R*- invariant, *S* - while condition, *C*- while body, *Q*- post-condition):

1. by assumption $\vdash P \Rightarrow R$, formula *R* cannot be weaker than *P* i.e., possibly *P* includes in its conjuncts propositions of *R*, i.e., *R* must include at most those variables that occur in *P*.
2. by assumption $\vdash R \wedge \neg S \Rightarrow Q$, *R* includes conjuncts that strengthen the condition $\neg S$ enough to imply *Q*, i.e., *R* must include at least all those variables of *Q* that do not occur in *S*.
3. by assumption $\vdash \{R \wedge S\}\ C\ \{R\}$ execution of *C* does not influence the validity of *R*, i.e., *R* is a "sort of balance equation" on variables and constants of *C*. Also variables of the rest of whole program can be referred in *R* as constants for *C*.
4. since for provability of (1) weak as much as possible *R* and for provability of (2) strong as much as possible *R* is preferable then (1) and (2) together bound the set of conjuncts of *R* from below and from above, so that $P \Rightarrow R$ and $(R \wedge \neg S) \Rightarrow Q$.
5. Analyzing the effect of *C*, one can find variables monotonously increasing ~~and~~ or decreasing when executing *C*, e.g.,

   (5.1) let $E_1$ and $E_2$ be expressions (defined using variables of *C*) increasing when *C* is executed iteratively. Then *R* may have a form of equation $f_1(E_1) = f_2(E_2)$ where $f_1$ and $f_2$ may be just simple multiplications with some constants to balance $E_1$ and $E_2$.

   (5.2) let $E_1$ be an expressions increasing and $E_2$ expression decreasing when *C* is executed iteratively and $f_3$ describing the final result of iteration. Then *R* may have the form $f_1(E_1)$ ¤ $f_2(E_2) = f_3$ where ¤ is multiplication or adding.

To practice with finding invariants, it is recommendable to write while-programs that compute factorial, Fibonacci numbers, and multiplication using only summation, also programs of array operations will do.

**Examples of invariants**:

**Example 1**:

```
{M≥1}
  BEGIN
        X:=0;
        FOR N:=1 UNTIL  M DO
          X:=X+N
  END
{X=(M×(M+1)) DIV 2}
```

Invariant: $\boxed{R \equiv X = N * (N - 1) \text{ DIV } 2 \wedge N \leq M+1}$

**Example 2:**

```
{T}
    BEGIN
          R:=X;
          Q:=0;
          WHILE Y ≤ R DO
            BEGIN
              R := R-Y; Q:= Q+1
            END
    END
{ X = R+Y×Q /\ R < Y }
```

Invariant: $\boxed{X = Y \times Q + R}$