In this exercise sheet we work with the automated verification techniques CEGAR (Predicate Abstraction) and Trace Abstraction.

Submit your solution by uploading it as PDF in ILIAS.

Exercise 1: 5 Points
Apply the CEGAR approach to the program below. Whenever you have to provide a sequence of statements you may return any sequence, but we encourage you to take the shortest sequence. Document all intermediate steps.

**Hint:** If you choose the abstraction of traces wisely, then two iteration steps are sufficient.

```
x := 0;
while (x <= 100) {
y := true;
x := x + 1;
}
assert y == true;
```

Exercise 2: Abstraction of a Trace 2 Points
In the lecture we defined an abstraction $\pi^#$ of a trace $\pi$, derived by replacing some of the statements $st$ with their abstract counterpart $\text{abstract}(st)$. The intuition is that sometimes a few statements in $\pi$ are sufficient to make it infeasible. A proof of infeasibility of $\pi^#$ is then also a proof of infeasibility of $\pi$.

In this exercise, we consider a modified concept of abstraction: Instead of replacing assignments with their abstraction (havoc), we delete them from the trace entirely.

Show that this is not a good notion of abstraction. In particular, give a trace $\pi$ and a corresponding abstraction $\pi^#$, such that $\pi^#$ is infeasible, but $\pi$ is feasible. Give a proof of infeasibility for $\pi^#$, and an execution for $\pi$. 
Exercise 3: Trace Abstraction

Consider the following control-flow graph for a program $P$, and let $A_P$ be the corresponding automaton. In this task, you should apply trace abstraction to prove that the program $P$ is safe.

Give two error traces $\pi_1$ and $\pi_2$ and construct corresponding Floyd-Hoare automata $A_1$ and $A_2$ such that the inclusion $L(A_P) \subseteq L(A_1) \cup L(A_2)$ holds.