Exercise 1: Termination
In the lecture, we discussed four different properties of programs. One property was termination the other properties where related to termination. We provide formal definitions here. In each case, we consider a program $P$ with a CFG $(\text{Loc}, \Delta, \ell_{\text{init}}, \ell_{\text{ex}})$.

(a) We say that $P$ can reach the exit location if there exists a finite execution, such that the first configuration $(\ell, s)$ is initial, and the last configuration is $(\ell_{\text{ex}}, s')$ for some state $s'$.

(b) We say that $P$ can stop if there exists a reachable configuration $(\ell, s)$ such that there exists no configuration $(\ell', s')$ and statement $st$ with $(\ell, st, \ell') \in \Delta$ and $(s, s') \in [st]$.

(c) We say that $P$ always reaches the exit location if there exist no infinite executions, and all finite executions end in a configuration $(\ell', s')$ where we either have a successor (i.e., there exists a configuration $(\ell'', s'')$ and statement $st$ with $(\ell', st, \ell'') \in \Delta$ and $(s', s'') \in [st]$) or we have that $\ell'$ is $\ell_{\text{ex}}$.

(d) We say that $P$ always stops (resp. $P$ terminates) if there exist no infinite executions.

In this exercise, you should give programs that differentiate between these definitions. In particular, for each of the following pairs, give a program such that one definition holds but the other does not. Explain which of the definitions holds and why.

(a) $P$ can reach the exit location vs. $P$ can stop
(b) $P$ can stop vs. $P$ always stops

Exercise 2: Ranking Functions
For each of the following programs, state whether it (always) terminates or not. If it terminates, give a ranking function for each loop in the program. If it may not terminate, give an infinite execution of the program.

```
   while (x > 0) {
     while (y > 0) {
       y := y-1;
     }
     x := x-1;
     havoc y;
   }
```

```
   while (x > 0) {
     if (y > 0) {
       y := y-1;
     } else {
       x := x-1;
     }
     havoc y;
   }
```

```
   while (x > 0) {
     if (y > 0) {
       y := y-1;
     } else {
       x := x-1;
     }
     havoc x;
   }
```

Listing 1: Program $P_1$
Listing 2: Program $P_2$
Listing 3: Program $P_3$
**Hint:** For simple loops is often convenient to use a function whose range is \( \mathbb{N} \) and the strictly greater than relation \( > \) on natural numbers. For more complex loops, this is sometimes not sufficient but we can use instead a function \( f : S_{V,\mu} \rightarrow \mathbb{N}_1 \times \ldots \times \mathbb{N}_n \) whose range are \( n \)-tuples of natural numbers and the **lexicographic order** \( >_{\text{lex}} \) that we define as follows.

\[
(m_1, \ldots, m_n) >_{\text{lex}} (m'_1, \ldots, m'_n) \iff \text{there exists } i \in \{1, \ldots, n\} \text{ such that } m_i > m'_i \\
\text{and for all } k \in \{1, \ldots, i-1\} \text{ the equality } m_k = m'_k \text{ holds}
\]

If a function with that signature together with the order \( >_{\text{lex}} \) is a ranking function, it is often called a **lexicographic ranking function**.

**Exercise 3: Synthesis of Ranking Functions** 2+2 Points
In this exercise we want to synthesize a ranking function for the program below and we want to use Farkas’ Lemma in order to simplify the constraints for the SMT solver.

```plaintext
while (7*x+5*y>=4 && 7*x+3*y<=-12) {
    x := 2*x - y;
    y := 3*y + 2;
}
```

(a) Construct a transition formula for the following statement.

\[\text{assume}(7*x+5*y>4 \land \land 7*x+3*y<-12); \ x:=2*x-y; \ y:=3*y+2;\]

(b) Write down the constraints (formula universally quantified over \( x, y, x', y' \)) that state that the loop has a ranking function \( f \) of the form \( f(x, y) = \alpha \cdot x + \beta \cdot y + \gamma \).

The symbols \( \alpha, \beta \) and \( \gamma \) are the coefficients of the ranking function for which we want to find a solution.

(c) Use Farkas’ Lemma to transform the constraints into logically equivalent, existentially quantified, constraints that use only linear arithmetic (e.g., no multiplication of variables). This is a bonus exercise and you may submit your solution as an SMT script.

(d) Find a solution for \( \alpha, \beta \) and \( \gamma \). This is also a bonus exercise.

**Exercise 4: Ranking Function** 2 Points
Let us consider the program whose code and control-flow graph are given below.

```plaintext
assume(y >= 1);
while (x >= 0) {
    x := x - y;
}
```

The program is terminating, however there is no ranking function for the while loop. To prove termination, we introduced the following definitions in the lecture:
Definition (Loop Entry) Given a while loop \texttt{while(expr)}\{\texttt{st}\} and a control-flow graph $G = (\text{Loc}, \Delta, \ell_{\text{init}}, \ell_{\text{ex}})$ for this while loop, we call $\ell_{\text{init}}$ the entry location of the while loop.

Definition (Ranking Function) Given a program $P = (V, \mu, st)$, a Floyd-Hoare annotation $\beta$ for $P$, a while loop \texttt{while(expr)}\{\texttt{st}\} whose loop entry is the location $\ell$, and a set $W$ together with a well-founded relation $R \subseteq W \times W$, we call a function $f : \mathcal{S}_{V,\mu} \rightarrow W$ a ranking function for \texttt{while(expr)}\{\texttt{st}\} and $\beta$ if for each pairs of states where $s \in \{\beta(\ell)\}$ and $(s, s') \in [\text{assume expr; st}]$, the relation $(f(s), f(s')) \in R$ holds.

Give a Floyd-Hoare annotation $\beta$ and a function $f$ such that $f$ is a ranking function for $\beta$ and the while loop.