Let us consider the following program.

```plaintext
1 assume (y >= 1);
2 while (x >= 0) {
3     x := x - y;
4 }
```

The program is terminating, however there is no ranking function for this while loop.

**Definition (Loop Entry)** Given a while loop `while(expr) {st}` and a control-flow graph `G = (Loc, Δ, ℓ_{init}, ℓ_{ex})` for this while loop, we call `ℓ_{init}` the *entry location* of the while loop.

**Definition (Ranking Function)** Given a program `P = (V, µ, st)`, a Floyd-Hoare annotation `β` for `P` a while loop `while(expr) {st}` whose loop entry is the location `ℓ` and a set `W` together with a well-founded relation `R ⊆ W × W` we call a function `f : Sv,µ → W` a *ranking function* for `while(expr) {st}` and `β` if for each pairs of states where `s ∈ {β(ℓ)}` and `(s, s') ∈ [assume expr; st]` the relation `(f(s), f(s')) ∈ R` holds.

**Exercise 1: Ranking Function**

2 Points

Give a Floyd-Hoare `β` and a function `f` such that `f` is a ranking function for `β` and the while loop.

**Exercise 2: Easter Bunny**

2 Bonus Points

Let `k` be an integer variable and `a` be an array that has integer indices and integer values. Is the following program terminating?

```plaintext
1 while (a[a[k]] >= 0) {
2     a[a[k]] := a[a[k]] - 1;
3 }
```

A story that motivates this program:
The array `a` models a street with infinitely many farmhouses. Each farmhouse has a finite number of eggs. Because of the zombie apocalypse no new eggs can be produced. The array `a` maps house numbers to the number of eggs that are available in the house. The easter bunny is iteratively stealing eggs (one at a time) according to the strategy `a[a[k]] := a[a[k]] - 1` until he visits a farm that ran out of eggs. Is this a bad strategy because the easter bunny will eventually visit always the same farm? Or is there some street and some choice of `k` in which the easter bunny can continue forever?