Exercise 1: Assume Statement 2 Bonus Points
In the lecture we introduced the assume expr; statement in Boostan, and we formally defined its semantics.

However, there already exists a Boostan statement with the same semantics, i.e., a statement st such that \([st] = [\text{assume expr;}]\), and such that no assume statement occurs in the statement st.

(a) Give such a statement st.

(b) Since st has the same semantics as the statement assume expr;, they satisfy exactly the same precondition-postcondition pairs.

However, in another sense, the two statements behave quite differently. Explain how, and in which context this is relevant.

Exercise 2: Hoare Proof System with Havoc and Assume 2 Points
Provide a Hoare logic proof that shows that the following Boostan program P satisfies the precondition-postcondition pair \((\{x > 0\}, \{x > 0\})\).

\[
\text{havoc y; assume } x > y; x := x - y;
\]

Exercise 3: Alternative Assume Axiom 2 Points
In the lecture, we introduced the following axiom for the assume statement in the Hoare proof system:

\[
\{\varphi\} \text{assume expr; } \{\varphi \land \text{expr}\} \quad (\text{assu})
\]

Alternatively, we could have introduced the following axiom:

\[
\{\text{expr } \rightarrow \psi\} \text{assume expr; } \{\psi\} \quad (\text{assu'})
\]

In this exercise we will show that both rules are equivalent.

(a) Give a proof for the Hoare triple \(\{\text{expr } \rightarrow \psi\} \text{assume expr; } \{\psi\}\) (for an arbitrary formula \(\psi\)) using the Hoare proof system.

(b) Give a proof of the Hoare triple \(\{\varphi\} \text{assume expr; } \{\varphi \land \text{expr}\}\) for an arbitrary formula \(\varphi\). Use a modified variant of the Hoare proof system, where the rule (assu) has been replaced by the rule (assu').
Exercise 4: Minimum  
2 Points
The following Boogie program iterates through a two-dimensional array and finds the minimum value within the given bounds \( lo \) and \( hi \).

```boogie
procedure findmin(a : [int, int]int, lo : int, hi : int) returns (min : int)
  requires lo <= hi;
  ensures (forall i, j : int :: lo <= i && i <= hi && lo <= j && j <= hi
  ==> a[i, j] >= min);
{ var i, j : int;
  i := lo;
  min := a[lo, lo];
  while (i <= hi) {
    j := lo;
    while (j <= hi) {
      if (a[i, j] < min) {
        min := a[i, j];
      }
      j := j + 1;
    }
    i := i + 1;
  }
}
```

Find inductive loop invariants for the two while loops of the program that are strong enough to prove that the program satisfies the given precondition-postcondition pair (the formulas after \texttt{requires} and \texttt{ensures}, respectively). You can use Ultimate Referee to check your solution.

Exercise 5: Bubble Sort  
2 Points
The following Boogie procedure implements the bubble sort algorithm that sorts a given array in ascending order.

```boogie
procedure BubbleSort(lo : int, hi : int, a : [int]int) returns (ar : [int]int)
  requires true;
  ensures (forall i, j : int :: (lo <= i && i <= j && j <= hi) ==> ar[i] <= ar[j]);
{ var i, j, t : int;
  ar := a;
  i := hi;
  while (i > lo) {
    j := lo;
    while (j < i) {
      if (ar[j] > ar[j+1]) {
        t := ar[j];
        ar[j] := ar[j+1];
        ar[j+1] := t;
      }
      j := j + 1;
    }
    i := i - 1;
  }
}
```

Find inductive loop invariants for the two while loops that are strong enough to prove that the program satisfies the given precondition-postcondition pair. You may also use your own solution to exercise 2 on exercise sheet 12 instead. You can use Ultimate Referee to check your solution.