Exercise 1: Havoc and Assume

Provide a Hoare logic proof that shows that the following Boo program $P$ satisfies the precondition-postcondition pair ($\{x > 0\}, \{x > 0\}$).

```
havoc $y$
assume $x > y$
x := $x - y$
```

Exercise 2: If-Then-Else with Havoc and Assume

Consider a program $P = (V, \mu, T)$ whose set of variables contains a boolean variable $b$, i.e., $b \in V$ and $\mu(b) = \{\text{true}, \text{false}\}$.

Let $st_1$ and $st_2$ be two statements of that program and let $st_3$ and $st_4$ be two statements that we define as follows.

- $st_3$: Havoc $b$; if(expr) {$st_1$} else {$st_2$}
- $st_4$: Havoc $b$; if($b$) {Havoc $b$; Assume expr; $st_1$} else {Havoc $b$; Assume $\neg$expr; $st_2$}

Show that the statements $st_3$ and $st_4$ are equivalent in the sense that we assign to both the same relation over program states, i.e., $[st_3] = [st_4]$.

Exercise 3: Square

Find inductive loop invariants for the while loop of the following program that is strong enough to prove that the program satisfies the given precondition-postcondition pair (the formulas after requires and ensures, respectively). Use Ultimate Referee\(^1\) to check your solution. Note that after the loop not only $res \geq 2 \cdot n$ but also $res = n \cdot n$ holds.

```
procedure main(n: int) returns (res: int)
requires n > 2;
ensures res >= 2*n;
{
    var i, odd : int;
i := 0;
odd := 1;
res := 0;
while (i < n) {
    res := res + odd;
    odd := odd + 2;
i := i + 1;
}
}
```

\(^1\)https://ultimate.informatik.uni-freiburg.de/?ui=int&tool=referee
Exercise 4: Minimum
The following Boogie program iterates through a two-dimensional array and finds the minimum value within the given bounds lo and hi.

```
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 requires and ensures, respectively). You can use Ultimate Referee to check your solution.

Exercise 5: Selection Sort
The following boogie procedure implements the selection sort algorithm that sorts a given array in ascending order.

```
procedure SelectionSort(lo : int, hi : int, a : [int]int) returns (ar : [int]int)
  requires lo <= hi;
  ensures (forall i1, i2 : int :: lo <= i1 && i1 <= i2 && i2 <= hi
    ==> ar[i1] <= ar[i2]);
{
  var i, k, min, tmp : int;
  ar := a;
  k := lo;
  while (k <= hi) {
    // Find the index of the minimal element between k and hi (inclusive)
    min := k;
    i := k + 1;
    while (i <= hi) {
      if (ar[i] < ar[min]) { min := i; }
      i := i + 1;
    }
    // Swap ar[k] and ar[min]
    tmp := ar[k];
    ar[k] := ar[min];
    ar[min] := tmp;
    k := k + 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 can use Ultimate
Referee to check your solution.