# Homework Assignment #6
# Due: March 13, 2023 at 10:00 p.m.

**1.** Consider an asynchronous system of $n$ processes, where processes may experience halting failures.

We can define the `1-2-Counter` type as follows. The state of the object stores a natural number. It provides three operations: READ returns the state of the object without changing it, INC increases the state of the object by 1 and returns ack, and INC-BY-2 increases the state of the object by 2 and returns ack.

[3] **(a)** Here is a proposed implementation of a `1-2-Counter` for $n$ processes from $n$ shared `read/write registers`, $A[1], \ldots, A[n]$. Process $i$ would execute the following code to perform an operation on the `1-2-counter`. (Here, $x$ and $v$ are local variables of the process performing the operation.)

```
1: function READ
2:     v ← 0
3:     for j ← 1 to n do
4:         v ← v + A[j]          ▷ This is a read of register A[j]
5:     end forreturn v
6: end function
7:
8: function INC
9:     x ← A[i]                  ▷ This is a read of register A[i]
10:    A[i] ← x + 1              ▷ This is a write to register A[i]
11: end function
12:
13: function INC-BY-2
14:    x ← A[i]                  ▷ This is a read of register A[i]
15:    A[i] ← x + 2              ▷ This is a write to register A[i]
16: end function
```

Prove this is *not* a linearizable implementation.

[3] **(b)** Is there a deterministic, wait-free, linearizable implementation of a `1-2-counter` from read/write registers? Prove your answer is correct.

[4] **(c)** Is there a deterministic, lock-free, linearizable, *anonymous* implementation of a `1-2-counter` from registers? Here, anonymous means that processes have no unique ids and all processes have identical code for each operation. Prove your answer is correct.

Hint: think about what happens when two processes trying to do the same operation run at exactly the same speed.