York University COSC6117 February 18, 2002

Homework Exercise
Due: March 4, 2002

5. We can define the `1-2-Counter` type as follows. The state of the object stores a natural number. It is equipped with 3 operations: READ returns the state of the object without changing it, INC increases the state of the object by 1 and returns ack, INC-BY-2 increases the state of the object by 2 and returns ack.

(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.)

READ
  $v \leftarrow 0$
  for $j \leftarrow 1$ to n
    $v \leftarrow v + A[j]$ % this is a read of `register` $A[j]$
  end for
  return v
end READ

INC
  $x \leftarrow A[i]$ % This is a read of `register` $A[i]$
  $A[i] \leftarrow x + 1$ % This is a write to `register` $A[i]$
end INC

INC-BY-2
  $x \leftarrow A[i]$ % This is a read of `register` $A[i]$
  $A[i] \leftarrow x + 2$ % This is a write to `register` $A[i]$
end INC-BY-2

Is this a linearizable implementation of a `1-2-Counter`? If so, prove it. If not, give a history that is not linearizable.

(b) What is the consensus number of the `1-2-Counter` type?

(c) What would the consensus number of the `1-2-Counter` type be if, instead of returning ack, the INC and INC-BY-2 operations returned the updated value of the counter?