Concurrent Algorithms October 27, 2019

Exercise 4

Problem 1. Write a wait-free algorithm that implements a *fetch-and-increment* object using atomic registers and compare-and-swap objects.

Reminder: Fetch-and-increment is a shared object that maintains a single variable c, initialized to 0, and provides a single operation *fetch&inc* with the following sequential specification:

```
operation fetch&inc()
  c' := c
  c := c + 1
  return c'
end
```

A compare-and-swap object is a shared object that maintains a single variable v, initialized to $\perp$, and provides a single operation *CAS* with the following sequential specification:

```
operation CAS(oldVal, newVal)
  v' := v
  if v = oldVal then v := newVal
  return v'
end
```