Further Modula-3

The thread system in Modula-3 uses *mutexes* and *condition variables* to control concurrency. An alternative scheme would be to provide *eventcounts* and *sequencers*. An eventcount is an integer, initially zero, equipped with the three atomic operations:

- **advance** increments the count and returns its new value,
- **read** returns the current value of the count, and
- **await (value)** suspends the calling thread until the count is at least as large as the value given as an argument.

A sequencer is an integer, initially zero, equipped with a single atomic operation:

- **ticket** increments the count and returns its previous value.

Given an eventcount, **guard**, and a sequencer, **turn**, a critical region can then be coded as follows:

```modula-3
myturn := turn.ticket ()
guard.await (myturn);

protected code

EVAL guard.advance ()
```

Write an interface, **ECS**, defining opaque object types **EventCount** and **Sequencer**. **EventCount** should have methods **advance**, **read** and **await**, with appropriate signatures, and **Sequencer** should have a **ticket** method. 

Sketch an implementation of the **ECS** module giving concrete revelations of the types and providing appropriate default methods.