Massachusetts Institute of Technology
Department of Electrical Engineering and Computer Science

6.826 – Principles of Computer Systems

Handout 11 September 20, 1993

Problem Set #2

Due date: September 27, 1993

Problem 1.

Consider the following specification of a multiset (also sometimes known as a "bag") module:

```
MODULE Multiset [T] =

TYPE S = T->Int

VAR s := S{* -> 0}

APROC Insert (t) =  << s(t) := s(t)+1 >>

APROC Delete (t) RAISES {NotFound} =
          << s(t) = 0 => RAISE NotFound
                     [*] s(t) := s(t)-1 >>

FUNC Member(t)->Bool = RET( s(t)>0 )

END Multiset
```

(A) Describe the state machine denoted by the specification above (*i.e.*, describe briefly the state domain, the initial states, and the state transitions.)

(B) Write an implementation of the Multiset module using a linear sequence with linear search to implement the multiset.

(C) Describe the state machine denoted by the implementation you wrote in part (B).

(D) Give an abstraction function and rep invariant for your implementation from part (B).

(E) Sketch a proof of correctness for your implementation from part (B), using your abstraction function and rep invariant from part (D). (List the cases that need to be considered, and for each case give a brief informal argument that the appropriate conditions are satisfied.)

Problem 2.

This problem was also used last year. Do not look at last year's solution; work it out on your own.

Prove the following theorem:

Theorem: Let I be an implementation, S be a specification, and AF be an abstraction function from I to S that satisfies the two conditions given in Handout 8. Then every trace of I is a trace of S.

Problem 3.

The most popular way to get a highly available disk is to use two physical disks for each logical one, and to write an identical copy of each block on both disks (this is called "mirroring" or "shadowing"). This approach protects both against the corruption of individual blocks and the failure, temporary or permanent, of an entire disk. We would like to maintain the invariant that for each block, both copies are the same or one of the copies gives an error when read, but this isn't possible, since the only atomic operation is to write a single disk block to one disk.

(A) Modify the Dsk spec given in Handout 9 to reflect the possibility that a disk block can have either permanent or transient read errors. Explain your model for how disks can fail and why your spec is consistent with your model.

(B) Give a precise statement of the invariant mentioned above as a predicate in Spec.

(C) Give an implementation of your Dsk spec using two copies of itself; in other words; give an implementation of a mirrored disk. Your implementation should maintain the invariant except while a Write is in progress. The tricky part of this is dealing with a failure during a write, either of a disk or of the process implementing the mirroring.

You should assume that if the process fails it stops executing after some atomic action, and the the next thing that happens is an invocation of a PROC called Crash, which must re-establish the invariant.