Computer Science 2221	**Homework 3**
University of Toronto	**Due: Friday, November 7, 2002**

1. The proof of the FLP impossibility result is based on the following two key facts about *any* (assumed) 1-tolerant Consensus algorithm A for the asynchronous model:

(i) A has an initial bivalent configuration.

(ii) For any bivalent configuration C of A, and any event e that is applicable to C, there is a finite schedule S so that Se is applicable to C and $Se(C)$ is bivalent.

In this question we examine the validity of these facts for 0-tolerant Consensus algorithms, i.e., algorithms that work assuming no failures occur. For each part below, clearly (and succinctly) justify your answer.

(a) Give a specific 0-tolerant Consensus algorithm that has no bivalent initial configuration.

(b) Give a specific 0-tolerant Consensus algorithm that has a bivalent initial configuration.

(c) What does each of (a) and (b) imply about the validity of Facts (i) and (ii) for 0-tolerant Consensus algoriothms?

2. The *asynchronous model with initial crashes* is the following restriction of the standard asynchronous model with processes crashes: A faulty process takes no steps at all, i.e., it is "dead" from the start. Thus in this model, if a process takes one step, it is guaranteed to be nonfaulty. As with the usual model we assume that links are reliable: A message sent to a nonfaulty process will eventually be received.

For any positive integers n, t such that $n > 2t$, give a t-tolerant Consensus algorithm for n processes in the asynchronous model with initial failures. (Hint: In the asynchronous model with initial failures, implement a failure detector that is at least as strong as $\diamondsuit\mathcal{S}$.)

3. Consider Ben Or's randomized Consensus algorithm. As given in class (and in the handout), this algorithm works for asynchronous systems with n processes such that up to $t < n/2$ of them may fail by general-omission failures.

(a) We proved that if $n > 2t$ then Ben Or's algorithm satisfies *Validity*, namely, if a correct process decides v then v is the initial value of some process.

Show that:

(i) If $n > 4t$, this algorithm satisfies *Strong Validity*, namely, if a correct process decides v then v is the initial value of some *correct* process.

(ii) If $n \leq 4t$, this algorithm does not satisfy Strong Validity. (Show this by describing an execution in which the decision value is not the initial value of a correct process.)

(b) If we assume that $n > 5t$, then with a simple modification Ben Or's algorithm works even for t *malicious* processes! The modification consists of changing the thresholds for sending a (P, k, v) message, for deciding v, and for setting $x_p := v$ as shown below.

$x_p :=$ initial value proposed by p (0 or 1)
for $k := 1, 2, ...$ **do**

send (R, k, x_p) **to** all processes
wait for messages of the form $(R, k, *)$ **from** $n - t$ processes
if received at least $(n + t + 1)/2$ (R, k, v) with the same v
then send (P, k, v) **to** all processes
else send $(P, k, ?)$ **to** all processes

wait for messages of the form $(P, k, *)$ from $n - t$ processes
if received at least $3t + 1$ (P, k, v) with $v \neq ?$
then decide v
if received at least $t + 1$ (P, k, v) with $v \neq ?$
then $x_p := v$
else $x_p :=$ 0 or 1, each with probability 1/2
od

Prove that the above algorithm indeed works (i.e., satisfies validity, agreement, and termination with probability 1) in a system with $n > 5t$ processes, t of which may be malicious.