Computer Science 2221	**Homework 4**
University of Toronto	**Due: Friday December 5, 2003**

Question 1. (20 marks) In the paper "A simple bivalency proof that t-resilient consensus requires $t+1$ rounds" (available through the course web page), the following result is shown:

> **Theorem.** *Consider a synchronous round-based system $\mathcal{S}$ with n processes and at most t crash failures* ***such that at most one process crashes in each round****. If $n > t+1$ then there is no algorithm that solves consensus in t rounds in $\mathcal{S}$.*

(a) Prove that the same theorem holds if we replace the word "consensus" by "weak consensus", where the weak consensus problem is the weaker version of consensus with the validity property replaced by

- ***Nontriviality:*** There is an execution in which a correct process decides 0 and there is an execution in which a correct process decides 1.

You need only describe in detail those parts of your proof that are different from the proof given in the above-cited paper.

(b) Consider a synchronous round-based system $\mathcal{S}'$ with n processes and at most t crash failures. Unlike the system $\mathcal{S}$ in the previously stated theorem, in $\mathcal{S}'$ we do ***not*** restrict failures to occur at most one per round.

(i) Give a 2-round algorithm that solves weak consensus in $\mathcal{S}'$.

(ii) Is there a 2-round algorithm that solves consensus in $\mathcal{S}'$? Justify your answer.

Question 2. (20 marks) Consider a variation of TRB known as the ***crusader agreement problem***. In this problem, there are n processes, one of which is a sender with a message it wants to broadcast (different from the special value **SF**). Of the n processes, at most t may fail and the others are correct. The problem is to come up with an algorithm that achieves the following properties:

- ***Termination:*** Each correct process delivers a message.
- ***Validity:*** If the sender is correct, then all correct processes deliver the sender's message.
- ***Weak agreement:*** If two correct processes deliver messages m and m', and neither m nor m' is **SF**, then $m = m'$.

Informally, when the sender is faulty, this problem allows some correct processes to deliver **SF** and other correct processes to deliver some non-**SF** message. Correct processes, however, may not deliver different non-**SF** messages.

(a) For the synchronous round model ***with arbitrary failures*** prove the following:

(i) For any n, t, if $n > 3t$, there is a two-round algorithm for the crusader agreement problem.

(ii) For any n, t, if $n > 3t$, there is no one-round algorithm for the crusader agreement problem.

(iii) For any n, t, if $n \leq 3t$, there is no algorithm for the crusader agreement problem (no matter how many rounds we allow).

(b) For the synchronous round model ***with arbitrary failures and message authentication***, give a 2-round algorithm for the crusader agreement problem that works for all n and t such that $n \geq t \geq 1$.

Question 3. (20 marks) Consider an asynchronous system $\mathcal{S}$ with crash failures. Let $\mathcal{A}$ be ***any*** algorithm that solves consensus in $\mathcal{S}$ when processes have access to the ***eventually perfect*** failure detector $\Diamond P$. (Recall that $\Diamond P$ satisfies the following properties: (i) eventually each crashed process is permanently suspected by each correct process; and (ii) there is a time after which no correct process is suspected by any correct process.)

(a) Sam says that $\mathcal{A}$ must solve ***uniform*** consensus. Is he right? Briefly justify your answer.

(b) Let D be a failure detector that returns an ***arbitrary*** list of suspects. Thus, we cannot make any assumption on the quality of D's output.

Suppose we execute algorithm $\mathcal{A}$ with failure detector D (instead of $\Diamond P$). Vassos says that $\mathcal{A}$ may now violate the agreement property of consensus. Is he right? Briefly justify your answer.

(c) Revisit the above two questions in case $\mathcal{A}$ is an arbitrary algorithm that solves consensus in $\mathcal{S}$ when processes have access to the ***perfect*** failure detector P (instead of the eventually perfect failure detector $\Diamond P$). (Recall that P satisfies the following two properties: (i) eventually each crashed process is permanently suspected by each correct process, and (ii) no process is suspected before it crashes.

Question 4. (15 marks) Let $\mathcal{S}$ be an asynchronous system with n processes, t of which may crash, and where links are reliable. Assume $n > 3t$. Consider the following implementation of a failure detector in $\mathcal{S}$.

- Every process p periodically:
  - (i) Sends an "are you alive?" message to every process in the system.
  - (ii) Waits to receive an "I am alive" message from $n - t$ distinct processes.
  - (iii) Suspects the t processes from which it did not receive the "I am alive" message; this list of suspects is used until the next period.
- Every process that receives "are you alive?" from a process q, replies with "I am alive" to q.

(a) Does this failure detector satisfy the ***strong completeness*** property? Briefly justify your answer.

(b) Does it satisfy the following property: At each time τ and for each correct process p, a majority of the correct processes is not suspected by p? Briefly justify your answer.

(c) Can one transform this failure detector into $\Diamond S$ in system $\mathcal{S}$? If so, give an algorithm that effects this transformation and justify its correctness. If not, explain why.

Question 5. (10 marks) A single-writer, multi-reader register R is ***less-than-safe*** if it satisfies the following property:

- If a read operation on R is not concurrent with ***any*** other operation on R then it returns the value last written into R.

A less-than-safe register is weaker than a safe register, where a read operation returns the value last written as long as it is not concurrent with any ***write*** operation on the register.

Lamport showed that it is possible to solve the mutual exclusion problem in an asynchronous shared-memory system using only safe registers. Is it possible to do so using only less-than-safe registers? Justify your answer.

Question 6. (10 marks) Shown below is a proposed mutual exclusion algorithm for the asynchronous model where processes communicate via shared registers with write and read operations. Prove or disprove:

(a) The algorithm satisfies mutual exclusion and deadlock freedom if the shared registers are atomic.

(b) The algorithm satisfies mutual exclusion and deadlock freedom if the shared registers are safe.

```
communication var:
    control_i: integer in range 0..2 initially 0
local var:
    j: integer in range 1..N

repeat
    NCS            % non-critical section %
L0: control_i := 1
L1: for j := 1 to i − 1 do
        if control_j ≠ 0 then goto L1
    control_i := 2
    for j := 1 to N do
        if j ≠ i and control_j = 2 then goto L0
    CS             % critical section %
    control_i := 0
forever
```