Note: YOU MUST DO THE HOMEWORK BY YOURSELF. If you have difficulties in solving a question you may discuss it with friends, BUT you MUST phrase, write and formulate the answers by yourself, after you understand the solution. The language of the solution must be entirely your own.

Notice 2: Unless your hand writing is extremely clear, you should type (Word, or LaTeX, etc.) and print your homework.

Notice 3: Put your name and id number on each page of the solution.

1. Processors in a shared memory concurrent system are equipped with the operation Memory-Memory-Copy which atomically copies one shared memory word to another shared memory word. I.e., it is given two addresses, $a1$ and $a2$, and it atomically copies the content of Memory[$a1$] into Memory[$a2$]:

$$\text{MM-copy}(a1, a2):\text{Memory}[a2] := \text{Memory}[a1];$$

What is the consensus number of this operation. That is, what is the maximum number of processors in such a system that can solve consensus when the only operations they may use are atomic read or write and MM-copy($a1$, $a2$).

2. Describe a leader election algorithm in a shared memory concurrent system that supports atomic read and write operations and atomic binary-consensus. Operations on an atomic $B\text{Consensus}[x]$ is invoked by any processor by a propose.$B\text{Consensus}[x]$(input: boolean) operation. In its sequential specification the operation returns the input of the first processor that performed propose on $B\text{Consensus}[x]$. The LE, leader election algorithm returns the id of a processor that has invoked the elect(my id) operation, to all the processors, i.e., it is like a multi valued consensus. Assume all processor ids are represented by a vector of 5 log $n$ bits.

3. (at most two pages) What is the consensus number of the following variants of Fetch and Add (F&A). You may assume any size (number of bits) in the Fetch_and_Add register. Also, $n$ the number of processes is known in advance. As part of the implementation the Fetch_and_Add register may be started (initialized) to any non-negative number.

   (a) Positive-cut F&A. Here is its sequential specification:
   (b) Absolute F&A. Here is its sequential specification:

4. In exclusion-terminating_with_default consensus task $T$ is invoked by calling $\text{propose}(input)$, which returns an output $o$ that can be either a value or default, such that:
Fetch&Add_Positive-cut (v, R); /* v Integer and R a register */
  tmp1 := read(R);
  tmp2 := max{0, tmp1 + v};
  R := tmp2;
  Return(tmp1)

Fetch&Add_Absolute (v, R); /* v Integer and R a register */
  tmp1 := read(R);
  tmp2 := |tmp1 + v|;
  R := tmp2;
  Return(tmp1)

Agreement: If an invocation of propose(input) returns o = v, v ≠ default then any other invocation of propose(input) returns o = v or o = default.

Validity: If an invocation of propose(input) returns o = v which is not default then there is an invocation of propose(v) by some process, which does not return default.

Exclusion-Termination with default: If during the entire invocation of propose(input) by some process p no other process took any step, then p returns o ≠ default (the other process may be after the invocation and before the response but not taking any steps, i.e., not accessing any of the primitive objects). Any other invocation of propose(input) eventually returns a value o which is either default or some value v.

(a) (try in 1 page, but at most 2 pages) Show that there is no implementation of such a task using read/write registers. Hint: show this impossibility by a reduction to wait-free consensus for two processes.

(b) (try in 1 page, but at most 2 pages) In this section we change the following condition: Exclusion-Termination with default condition is changed as follows: If process p completes its entire invocation of propose(input_p) before any other process takes any step then p returns o = input_p. Any other invocation of propose(input) eventually returns a value o which is either default or v. Is this task read/write implementable?

5. (at most 2 pages) Consider the approximate agreement problem, which is defined as an agreement problem as follows: The value domain V (of all possible inputs and outputs) is the set of rational numbers. Each processor starts with an input from V and exits with an output also from V such that: (a) every output is in the range of the inputs (i.e., between the minimum input and the maximum input), and (b) the difference between any two output values is at most one. Suppose we are given a 10-process, 2-fault-tolerant asynchronous shared memory algorithm A that solves approximate agreement, using only read/write atomic registers. Describe clearly how we can use algorithm A and the BG-simulation (Borowsky-Gafni) results to obtain
a 3-process wait-free asynchronous shared memory algorithm to solve approximate agreement, again using only read/write atomic registers. Recall that in class we used the BG-simulation to construct a wait-free 2 processors consensus algorithm from an $n > 2$ processors 1 fault-tolerant asynchronous shared memory consensus algorithm (a variant of which appears in Section 5.3.2 in the Attiya Welch book “distributed Computing”)
