CS-5300 PCP
Theory Assignment-3

NAME: Souvik Sarkar
ROLL: CS23MTECH02001

4.5) ANS:

False. Substituting the atomic SRSW (Single Reader, Single Writer) registers with regular, non-atomic SRSW registers in the design does not result in an atomic MRSW (Multiple Readers, Single Writer) register. In the provided setup, each element in the 'a table' functions as an atomic SRSW register. These atomic registers guarantee that read and write operations within each element occur atomically, meaning there is no interference from simultaneous operations. This atomic nature is crucial for the proper functioning of the MRSW register.

If we were to use regular, non-atomic SRSW registers instead of the atomic ones, there would be no assurance of atomicity within each element. Multiple readers might read and write to the same element concurrently, leading to potential race conditions and data inconsistencies. Without atomicity, it cannot be ensured that read and write operations will be executed in a manner that preserves the integrity of the MRSW register.

Hence, replacing the atomic SRSW registers with regular SRSW registers would undermine the atomic nature of the MRSW register, rendering it unsuitable for concurrent operations involving multiple readers and a single writer.

4.8) ANS:

The two atomic 32-bit memory locations (registers) used to build the 64-bit register do not meet the criteria for an "atomic register." Read and write operations in an atomic register should appear to be carried out instantly and sequentially without any interruption from other processes. This approach uses two independent 32-bit operations on various memory regions to read and write a 64-bit value. There is therefore no assurance that read and write operations will appear atomic to other processes or threads accessing the 64-bit register simultaneously. The fact that this 64-bit register is a "safe register" makes it the option that best fits the above criteria. Safe registers offer a type of consistency but not necessarily atomicity by guaranteeing that a read action returns the outcome of some writing operation. It does not, however, offer the complete atomicity needed for an atomic register.

5.12) ANS:

1. In the case of binary consensus, each thread simply puts its input to the bit, reads the bit, and selects the value it decoded.

2. Each thread in m-value consensus shares an array of $\log_2 m$ StickyBit objects and has an atomic register. Each thread tries to set each bit in the array to its corresponding bit in its input, starting with bit 0, after first writing its input to its public register. As long as it succeeds (the value in the array matches its input), it keeps going. When a thread loses, it switches from its own preference to one that matches the sticky bits as they have been set thus far in another thread.

5.22) ANS:

The issue is that the snapshot hides the queue's current state from you. Making room for the head before adding a value to it is not an atomic action. Begin with a clean queue. The first thread does an enqueue, moves the item up in the head, and increments the counter, but it does not write its value, leaving that slot empty. The value is then fully finished enqueuing by a second thread. After that, it takes a snapshot and dequeues the image. When it completes the for loop and discovers that the space the first thread had created is still empty, it assumes that the value of the second enqueue is at the front of the queue and chooses this value. The first thread then arises, completes writing its value, calls dequeue on a snapshot to see its value is first, decides on it, and so forth. Thus, agreement cannot be reached.