Problem 1 (0 Points)

a) Give an algorithm that uses the TestAndSet hardware mutual exclusion instruction to update a shared variable X in a consistent manner.

b) Suppose that there are 1,000 processes that potentially can update X, but only a few (2 or 3) concurrently want to update X. How does the TestAndSet instruction speed-up the updating of X compared to a software-only algorithm (e.g., Peterson’s algorithm)?

Problem 2 (0 Points)

Consider the following algorithm and assume that we have created N processes.

Shared Variables:

- Semaphore X = N, Y = 1;
- Semaphore Z[N] = 0; // array of N semaphores initialized to 0
- int n = 0, w = 0;

Process i:

```c
int next = (i+1) mod N;
do forever {
    Wait(X);
    ... Compute ...
    Wait(Y);
    n = n + 1;
    if (n >= N) {
        n = 0; // Place A
        w = i;
        Signal(Y);
        Signal(Z[next]);
    } else {
        Signal(Y);
        Wait (Z[i]);
        if (next != w) Signal(Z[next]);
    }
    Signal(X);
}
```

a) What is the purpose of each of the semaphores X, Y, and Z[i]?

b) What would be the effect of deleting the statement labeled Place A from the algorithm?