ASSIGNMENT 1: QUESTION 1

PART A – MUTUAL EXCLUSION

Circular buffer			
dataType array [0..N-1] buffer integer in, out ← 0			
p		**q**	
	dataType d		dataType d
	loop forever		loop forever
p1:	d ← getItem()	q1:	await in != out
p2:	await out != (in+1) mod N	q2:	d ← buffer[out]
p3:	buffer[in] ← d	q3:	out ← (out+1) mod N
p4:	in ← (in+1) mod N	q4:	useItem(d)

$(in = out) \wedge (p3 \wedge q2)$ – both processors are in their critical sections, and they are pointing to the same location in the buffer

$(p3 \wedge q2) \rightarrow (in \neq out)$ – invariant represents the mutual exclusion

$(in \neq out) \vee \neg(p3 \wedge q2)$ – mutual exclusion

The following are scenarios that we want to avoid ensuring mutual exclusion:

- when the two threads are in a particular program counter (p3 AND q2) AND their pointers are pointing to the same element in the buffer (in = out) the array must be empty, and the next operation can only be a read. We can represent this by the number of read and write operations which we will refer to as nRead (p3) and nWrite (q2). If the number of operations is the same (nRead = nWrite) then the array is empty.

$$(in = out) \leftrightarrow (nRead = nWrite)$$

- The write operation (q2) cannot overtake the read operation in the circular buffer. This ensures that the algorithm does not try to write from an empty array element and cause reading and writing from the same element in an array simultaneously.

$$nRead \geq nWrite$$

- The above two imply that if the pointers are pointing to separate elements in the array that the array is not empty and therefore the number of reads must be greater than the number of writes.

$$(in \neq out) \rightarrow nRead > nWrite$$

- Finally, the buffer is bound by length N. Therefore, the number of reading and writing operations cannot exceed the size of the buffer. Since the algorithm utilizes modulus, we can say that $nRead \bmod N < N$ and $nWrite \bmod N < N$. That means that the difference between nRead and nWrite cannot exceed N.

$$|nRead - nWrite| \leq N$$

LEMMA 1

$$A = (in = out) \leftrightarrow (nRead = nWrite)$$

PROOF

Base Case: Array is empty, therefore in and out point to the same location. Number of operations for read and write are both zero. $(in = out) \leftrightarrow (nRead = nWrite)$ is True initially.

Inductive Step:

- Executing p1 does not change the truth of A.
- Executing p2 does not change the truth of A.
- Executing p3 and p4 will increase the number of Read operations making $(nRead = nWrite)$ false and will move the 'in' pointer making $(in = out)$ false as well which does not change the truth of A.
- This will then allow q1 to execute as, q1 cannot execute whilst in and out are pointing to the same location, which does not change the truth of A.
- Executing q2 and q3 with increase the number of Write operations making $(nRead = nWrite)$ false and will move the 'out' pointer making $(in = out)$false as well which does not change the truth of A.
- Executing q4 does not change the truth of A.

LEMMA 2

$$B = nRead \geq nWrite$$

PROOF

Base Case: Array is empty, therefore the number of operations for read and write are both zero.
$nRead \geq nWrite$ is True initially.

Inductive Step:

- Executing p1 does not change the truth of B.
- Executing p2 does not change the truth of B.
- Executing p3 will increase the number of Read operations $(nRead \geq nWrite)$ will remain true which does not change the truth of B.
- Executing p4 does not change the truth of B.
- This will then allow q1 to execute as, q1 cannot execute whilst in and out are pointing to the same location, which does not change the truth of B.
- Executing q2 will increase the number of Write operations $(nRead \geq nWrite)$ will remain true which does not change the truth of B.
- Executing q3 does not change the truth of B.
- Executing q4 does not change the truth of B.

LEMMA 3

$$C = (in \neq out) \rightarrow nRead > nWrite$$

PROOF

Base Case: Array is empty, therefore in and out point to the same location. Number of operations for read and write are both zero. $(in \neq out) \rightarrow nRead > nWrite$ is True initially as $(in \neq out)$ and $(nRead > nWrite)$ are both initially false. $False \rightarrow False$ is True.

Inductive Step:

- Executing p1 will not change the truth of C.
- Executing p2 will not change the truth of C.
- Executing p3 and p4 will increase the number of Read operations making $(nRead > nWrite)$ true and will move the 'in' pointer making $(in \neq out)$ true as well which does not change the truth of C.
- This will then allow q1 to execute as, q1 cannot execute whilst in and out are pointing to the same location, which does not change the truth of C.
- Executing q2 and q3 with increase the number of Write operations making $(nRead = nWrite)$ true and will move the 'out' pointer making $(in \neq out)$ true as well which does not change the truth of C.
- Executing q4 does not change the truth of C.

LEMMA 4

$$D = |nRead - nWrite| \leq N$$

PROOF

Base Case: The size of the Array is set to N and remains unchanged. The Array is empty, therefore the number of operations for read and write are both zero. $|nRead - nWrite| \leq N$ is True initially.

Assume N is greater than 1.

Inductive Step:

- Executing p1 will not change the truth of D
- Executing p2 with not change the truth of D but will maintain that the In pointer does not overtake the out pointer, therefore if the number of operations exceeds the size of the array, p2 will not execute.
- Executing p3 will increase the number of Read operations ($|nRead - nWrite| \leq N$) will remain true which does not change the truth of B.
- Executing p4 with not change the truth of D
- This will then allow q1 to execute as, q1 cannot execute whilst in and out are pointing to the same location, which does not change the truth of B.
- Executing q2 will increase the number of Write operations ($|nRead - nWrite| \leq N$) will remain true which does not change the truth of D.
- Executing q3 does not change the truth of D.
- Executing q4 does not change the truth of D.

THEOREM 1

$$(in \neq out) \vee \neg(p3 \wedge q2)$$

PROOF

Base Case: $(in \neq out) \vee \neg(p3 \wedge q2)$ is true initially as $(in \neq out)$ is false and $\neg(p3 \wedge q2)$ is true.

Inductive Step:

- When $p3 \wedge q2$ is false, the only statements which can make it true are p2 and q1
- P2 executes only when $(out \neq (in + 1)\ mod\ N)$ is true which is the same as saying $(nWrite \neq nRead + 1)$, and by Lemma 2 we know that $nRead \geq nWrite$ therefore nWrite cannot be greater than nRead.
- Q1 executes only when $(in \neq out)$ is true, and by Lemma 3 we know $(in \neq out) \rightarrow nRead > nWrite$.

- By Lemma 1 we know that for $p3 \wedge q2$ to be true $(in = out) \leftrightarrow (nRead = nWrite)$ must be true which implies that the array is empty. We know by Lemma 3 that $|nRead - nWrite| \leq N$ which means that it is not possible for $(in = out) \leftrightarrow (nRead \neq nWrite)$ to be true as this would only be possible if $|nRead - nWrite| > N$.
- Hence $(in \neq out) \vee \neg(p3 \wedge q2)$ is always true

PART B – FREEDOM FROM STARVATION

INVARIANTS

- $(in = out) \vee (p3 \wedge q2)$

WANT TO PROVE

$$\square\big((p2 \Longrightarrow \diamond p3) \wedge (q1 \Longrightarrow \diamond q2)\big)$$

LEMMA 5

$$\square\Big(\big(p2 \wedge (out = in + 1 \; mod \; N)\big) \Longrightarrow \diamond(out \neq in + 1 \; mod \; N)\Big)$$

Proof:

LEMMA 6

$$\square\Big(\big(q1 \wedge (in = out)\big) \Longrightarrow \diamond(in \neq out)\Big)$$

Proof:

LEMMA 7

$$\square(p1 \Longrightarrow \diamond p2)$$

As getItem() may not terminate since there may be no items available

Proof:

THEOREM 2

$$\square\big((p2 \Longrightarrow \diamond p3) \wedge (q1 \Longrightarrow \diamond q2)\big)$$

Proof: