Lock => 0

```
While (true) {
a1  while (compare-and-swap (&lock, 1, 1) != 0) ;

                /* do nothing */
a2      /* CS */
a3      Lock = 0;

a4      /* RS */
        }
```

① Mutual Exclusion : Lock → 0

p0(a1), p0(a2) CS — 0 != 1 → F

p1(a1), p1(a2) CS

Not satisfied, it runs 2 process in the CS at the same time.

② progress : Lock → ∅, ∅, 0

p0(a1), p0(a2) CS, p0(a3), p0(a4) RS — satisfied

p1(a1), p1(a2) CS, p1(a3), p1(a4) RS

p1(a1), p0(a2) CS

③ Bounded waiting : Lock → ∅, 0

p0(a1) waiting

p1(a1), p1(a2) CS

Not satisfied, Because the second process entered it's CS, before p0.

Lock => 1

```
while (true) {
a1  while (compare_and_swap(
      &lock, 0, 1) != 0);

      /* do nothing */

a2  /* Critical Section */

a3  lock = 0;

a4  /* Remainder Section */

    }
```

- This algorithm will loop infinitely in the (a1) statement bc the value of the Lock will never change, and with it being 1, no process can enter.

- Doesn't satisfy any of the three requirements.

① Mutual Exclusion:

Lock => 1

po(a1), waiting

why? $1 \neq 0$ => return 1
lock expected

$1 != 0$

=> (T)

② progress:

po(a1), waiting

③ Bounded waiting:

po(a1), waiting

MAXI

waiting [4] = F, F, F, F

```
Lock => 0
  {
a1  waiting [i] = 1 ;
a2  key = 1 ;
a3  while (waiting [i] && key)
a4    key = test_and_set (&lock);
a5  waiting [i] = 0 ;
a6    /* Critical section */
a7  j = (i+1) % n ;
a8  while ((j != i) && (!waiting [j]))
a9    j = (j+1) % n ;
a10 if (j == i)
a11   lock = 0 ;
    else
a12   waiting [j] = 0 ;
a13 /* Remainder Section */
  } while (true)
```

- This algorithm satisfies Mutual Exclusion, and bounded waiting, But doesn't satisfy progress.

why not?

Because the loop will loop infinitely without the statement a8 becoming (F).

① Mutual Exclusion :

Lock → ∅, 1, 1 — 0: ~~F~~, 1: ~~F~~, 2: F, 3: F (T T)

key → ~~1~~, ∅, 1

T & T

po(a1), po(a2), po(a3), po(a4)

F & T

po(a3), po(a5), po(a6) CS

p1(a1), p1(a2), p1(a3), p1(a4)

waiting.

② progress :

Lock → ∅, 1 — 0: ~~F~~, 1: F, 2: F, 3: F (~~T~~ F)

key → ~~1~~, 0

T & T — T & F

po(a1), po(a2), po(a3), po(a4), po(a3)

po(a5), po(a6) CS, po(a7) j = 0+1 % 4 = 1

T & T — T & T

po(a8), po(a9) j = 1+1 % 4 = 2, po(a8),

T & T

po(a9) j = 2+1 % 4 = 3, po(a8), po(a9)

- The loop continues without the statement a8 being (F)

③ Bounded waiting :

Lock → ∅, ~~1~~, 1 — 0: ~~F~~, 1: ~~F~~, 2: F, 3: F (T T)

key → ~~1~~, ∅, ~~1~~, 1

T & T

po(a1), po(a2), po(a3), po(a4),

T & F

po(a3), po(a5) → waiting.

T & T

p1(a1), p1(a2), p1(a3)

T & T

p1(a4), p1(a3), p1(a4)

waiting