¿Qué es un semáforo?

Indicar si el siguiente multiprograma termina.
Suponga atomicidad línea a línea.
Inicialmente x=50.

```
while(0<x<100){        while (0<x<100) {
    x=x+1                  x=x-1
}                      }
```

Suponga atomicidad línea a línea.
Inicialmente x=0

```
while(true) {          while(true) {
    x=x+1                  x=x+1
    x=x-1                  x=x-1
}                      }
```

Inicialmente x=0.
**No suponga** atomicidad, es decir cada incremento o decremento es: leer la memoria, operar, escribir en la memoria.

```
while(true) {          while(true) {
    x=x+1                  x=x+1
    x=x-1                  x=x-1
}                      }
```

Indique que valores puede tomar x.

Suponga atomicidad línea a línea. La variable i es privada de la componente de la izquierda, la variable j es privada de la componente derecha, el arreglo a es compartido.
Inicialmente i=j=0 y a=[2,2,...,2].

```
while(i<N) {    while(j<N) {
    a[i]=0          a[j]=1
    i++             j++
}               }
```

Indique que valores del arreglo a son posibles a la salida.

Suponga atomicidad línea a línea. La variable i es privada de la componente de la izquierda, la variable j es privada de la componente derecha, el arreglo a es compartido.
Inicialmente i=j=0, a=[2, 2, ..., 2], el semáforo s=0.

```
while(i<N) {        while(j<N) {
    sem_wait(s)         a[j]=1
    a[i]=0              j++
    i++                 sem_post(s)
}                   }
```

Indique que valores del arreglo a son posibles a la salida.

Se tiene la siguiente implementación de locks.

```
typedef struct __lock_t {
  int flag;
} lock_t;
```

```
void init(lock_t *mutex) {
  // 0 -> disponible, 1 -> tomado
  mutex->flag = 0;
}
```

```
void lock(lock_t *mutex) {
  if (mutex->flag == 0) // TEST la bandera
    if (mutex->flag == 0) // reTEST de la bandera!
      mutex->flag = 1; // gané el CTF!!!!
}
```

```
void unlock(lock_t *mutex) {
  mutex->flag = 0; // devuelvo la bandera
}
```

Decir si funciona:

Suponga atomicidad línea a línea. La variable i es privada de la componente de la izquierda, la variable j es privada de la componente derecha, el arreglo a es compartido.
Inicialmente i=j=0, a=[2, 2, ..., 2], el semáforo s=1 y t=0.

```
while(i<N) {        while(j<N) {
    sem_wait(s)         sem_wait(t)
    a[i]=0              a[j]=1
    i++                 j++
    sem_post(t)         sem_post(s)
}                   }
```

Indique que valores del arreglo a son posibles a la salida.

Para el siguiente multiprograma, decir si el ***Invariante*** se cumple.

| Pre: $\texttt{barco}=\texttt{raton}=0 \wedge \texttt{s1}=1 \wedge 0<N$ | |
|---|---|
| B: while(true) {<br>1 wait(s1);<br>2 barco = barco + 1;<br>} | R: while(true) {<br>a raton = raton + 1;<br>b post(s1);<br>} |
| Inv: $\lvert\texttt{barco}-\texttt{raton}\rvert \leq N$ | |