

## Lista III de Computação Concorrente:

Gabriel da Fonseca Ottoboni Pinho - DRE 119043838

Rodrigo Delpreti de Siqueira - DRE 119022353

01/06/2021

### Questão 1

1. O carro vindo do Norte poderá iniciar a travessia, pois a ponte está livre (`n_sul = 0`).
2. Eles seguem em frente, pois a ponte está livre (`n_sul = 0`) e a mutex `e` foi liberada antes da travessia.
3. O carro do sul fica retido no `espera`, pois `n_norte > 0`.
4. Os carros do sul ficam retidos no `espera`, pois `n_norte > 0`.
5. Quando o último carro do norte atravessar a ponte, teremos `n_norte = 0` e `libera` será chamada, permitindo que os carros do sul continuem.
6. Sim, as variáveis `n_sul` e `n_norte` são protegidas pelo semáforo `e` e a variável `cont` é protegida por `em`.
7. Sim, caso a chegada de carros em uma direção seja maior que o tempo da travessia, a direção oposta nunca terá a oportunidade de atravessar.

## Questão 2

O erro ocorre quando o produtor produz enquanto o consumidor está executando `consome_item`. Se o item consumido for o último do buffer, `n` será incrementado pelo produtor durante o consumo, fazendo com que o semáforo `d` não seja resetado. Com isso, `d` terá valor 2, o que permite que um item que não existe seja retirado do buffer. Além disso, o `if` do consumidor utiliza a variável `n` sem exclusão mútua, o que pode causar outros problemas.

A solução é mover o `if` para “dentro” da mutex (semáforo `s`), no início do loop, de modo que a mutex é devolvida antes de `sem_wait`, mantendo a possibilidade de execução concorrente. A função `prod` permanece inalterada.

```
void *cons(void *args) {
    int item;
    while (1) {
        sem_wait(&s);
        if (n == 0) {
            sem_post(&s);
            sem_wait(&d);
            sem_wait(&s);
        }

        retira_item(&item);
        n--;
        sem_post(&s);

        consome_item(item);
    }
}
```

## Questão 3

1. Todos os semáforos devem ser iniciados com o valor 1. Como `em_e` e `em_l` fazem o papel de garantir a exclusão mútua das variáveis `e` e `l`, eles devem começar com 1. `escrita` e `leitura` bloqueiam as threads a fim de garantir os requisitos do padrão leitores e escritores. Para isso, `sem_wait(&escrita)` bloqueia novos escritores e o semáforo precisa começar com 1.
2. Ele consegue ler, pois `leitura` e `escrita` terão valor 1, permitindo que

ele não seja bloqueado.

3. Ele consegue ler, pois o leitor anterior terá setado **leitura** de volta para 1 antes de começar a ler, permitindo a concorrência.
4. Ele fica retido, pois o primeiro leitor setou **escrita** para 0 antes de começar a ler. O valor só retorna para 1 após o último leitor terminar sua tarefa.
5. Ele fica retido, pois o escritor setou **leitura** para 0 antes de se bloquear, garantindo a prioridade para escrita.
6. O último leitor a terminar vai setar **escrita** para 1, permitindo que o escritor trabalhe.
7. Nas linhas 3 e 5 da função dos leitores, não é necessário utilizar o semáforo **em\_1**, pois **leitura** já garante a exclusão mútua.