# Lista de Exercícios: Análise de Exclusão Mútua

### Instruções

Para cada exercício, analise o código apresentado e determine se ele satisfaz ou não os 4 critérios de exclusão mútua:

- 1. Exclusão Mútua: Apenas uma thread pode estar na região crítica simultaneamente
- 2. **Ausência de Atraso Desnecessário**: Uma thread sozinha deve poder entrar na RC imediatamente
- 3. Ausência de Deadlock: Pelo menos uma thread deve conseguir progredir
- 4. Ausência de Starvation: Todas as threads devem conseguir entrar na RC em tempo finito

### **Exercício 1**

```
// Variáveis globais
int contador = 0;
bool quer[2] = {false, false};
// Código das threads
void thread(int eu, int outro) {
  quer[eu] = true;
  contador++;
  while (contador > 1) {
 if (!quer[outro]) {
 contador--;
 quer[eu] = false;
   while (!quer[outro]);
    quer[eu] = true;
    contador++;
  // REGIÃO CRÍTICA
  contador--;
  quer[eu] = false;
```

```
// Variáveis globais
volatile int flag = 0; // 0 = livre, 1 = thread0, 2 = thread1

// Código das threads
void thread(int id) {
    int meu_valor = id + 1;
    ...
    while (flag != 0 && flag != meu_valor) {
        // espera
    }
    flag = meu_valor;
    // REGIÃO CRÍTICA

...
flag = 0;
}
```

# **Exercício 5**

```
// Variáveis globais
bool interessado[2] = {false, false};
bool prioridade[2] = {false, false};

// Código das threads
void thread(int eu, int outro) {
... interessado[eu] = true;
prioridade[eu] = true;
prioridade[outro] = false;

while (interessado[outro] && prioridade[outro] == false) {
... // espera
}

// REGIÃO CRÍTICA

interessado[eu] = false;
}
```

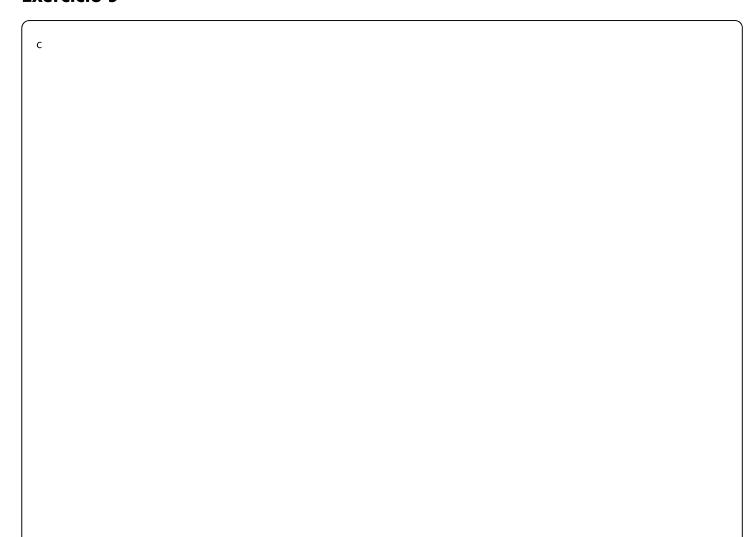
```
c

// Variáveis globais
int ticket = 0;
int vez = 0;

// Código das threads
void thread() {
... int meu_ticket = ticket++;
... ... while (vez != meu_ticket) {
... ... // espera
... }

// REGIÃO CRÍTICA
```

```
// Variáveis globais
bool fase1[2] = {false, false};
bool fase2[2] = {false, false};
int ultimo_fase1 = 0;
int ultimo_fase2 = 0;
// Código das threads
void thread(int eu, int outro) {
// Fase 1
  fase1[eu] = true;
  ultimo_fase1 = eu;
  while (fase1[outro] && ultimo_fase1 == eu) {
   // espera
  // Fase 2
  fase2[eu] = true;
  ultimo_fase2 = eu;
  while (fase2[outro] && ultimo_fase2 == eu) {
    // espera
  // REGIÃO CRÍTICA
  fase2[eu] = false;
  fase1[eu] = false;
```



```
// Variáveis globais
bool requisitando[2] = {false, false};
bool na_regiao[2] = {false, false};
// Código das threads
void thread(int eu, int outro) {
  requisitando[eu] = true;
  while (requisitando[outro]) {
  requisitando[eu] = false;
  while (na_regiao[outro]) {
       // espera
    requisitando[eu] = true;
  na_regiao[eu] = true;
  requisitando[eu] = false;
  // REGIÃO CRÍTICA
  na_regiao[eu] = false;
```

```
// Variáveis globais
volatile int turno = 0;
volatile bool ativo[2] = {false, false};
int contador_tentativas[2] = {0, 0};
// Código das threads
void thread(int eu, int outro) {
 _ativo[eu] = true;
  contador_tentativas[eu]++;
  if (contador_tentativas[eu] % 2 == 1) {
 turno = outro; // threads impares dão preferência
turno = eu; // threads pares tomam preferência
  while (ativo[outro] && turno == outro) {
    // espera
  // REGIÃO CRÍTICA
  ativo[eu] = false;
```