Nesse roteiro, sua equipe deve escolher 1 de 2 problemas para resolver:

1. **O problema do barbeiro dorminhoco**
   Na barbearia há um barbeiro, uma cadeira de barbeiro e n cadeiras para eventuais clientes esperarem a vez. Quando não há clientes, o barbeiro senta-se na cadeira de barbeiro e cai no sono. Quando chega um cliente, ele precisa acordar o barbeiro. Se outros clientes chegarem enquanto o barbeiro estiver cortando o cabelo de um cliente, eles se sentarão (se houver cadeiras vazias) ou sairão da barbearia (se todas as cadeiras estiverem ocupadas). O problema é programar o barbeiro e os clientes sem cair em condições de disputa (condições de corrida) (race conditions). Condições de disputa são situações em que duas ou mais threads (ou processos) estão trabalhando juntas e podem compartilhar algum armazenamento comum que cada uma pode ler e gravar. O resultado final depende de quem executa precisamente quando. Os resultados da maioria dos programas são bons, mas, de vez em quando, acontece algo estranho e inconsistente. Esse problema é semelhante a situações com várias filas, como uma mesa de atendimento de telemarketing com diversos atendentes e com um sistema computadorizado de chamadas em espera, atendendo a um número limitado de chamadas que chegam. Uma solução usa três semáforos: customers, que conta os clientes à espera de atendimento (exceto o cliente que está na cadeira de barbeiro, que não está esperando); barbers, o número de barbeiros (0 ou 1) que estão ociosos à espera de clientes, e mutex, que é usado para exclusão mútua. Precisamos ainda de uma variável, waiting, que também conta os clientes à espera de atendimento. É essencialmente uma cópia de customers. A razão de se ter waiting é que não há uma maneira de ler o valor atual do semáforo. Nessa solução, um cliente que entra na barbearia deve contar o número de clientes à espera de atendimento. Se este for menor que o número de cadeiras, ele ficará; do contrário, ele sairá.

2. **O problema do banheiro unissex:**
   Suponha que em um local público exista apenas um banheiro com **n** boxes (um compartimento com um sanitário). O banheiro pode ser usado tanto por homens e mulheres, mas não ao mesmo tempo.

   Faça um algoritmo concorrente baseado em monitor que controle o uso do banheiro.

   A entrada no banheiro é feita pelo procedimento *enterBathroom*. Depois de entrar no banheiro com sucesso, use o procedimento *getStall*, para usar um box. Se todos os boxes estiverem em uso então eles esperam em uma fila (não há espaço suficiente no banheiro). Depois de usar o box, cada pessoa avisa para que os outros usuários possam usá-lo.

   O programa deve ser justo no seguinte sentido. Suponha que, em um determinado instante no tempo, o banheiro está em uso por x pessoas de um mesmo sexo (alguns usando boxes e outros em espera), e a primeira pessoa do sexo oposto chega, chamada P.

   Então:

   - P entra no banheiro, logo depois da saída de todos o x indivíduos do banheiro,
   - Enquanto P está esperando, se outros indivíduos do mesmo sexo chegam, eles usarão o banheiro simultaneamente com P,

- Enquanto P está à espera, se indivíduos do sexo oposto chegam para usar o banheiro, eles entram no banheiro depois de P (e seus companheiros do mesmo sexo, se for o caso) saírem do banheiro,
- Enquanto P (e companheiros do mesmo sexo) estão usando o banheiro, se pessoas do mesmo sexo de P chegar, eles vão esperar todas as pessoas do sexo oposto de P saírem antes de começar a usar o banheiro.