Exercícios

01) **Atendimento no Bar**: O funcionamento do atendimento em um bar é baseado em um *bartender* que recebe o pedido dos garçons, que por sua vez recebe o pedido dos clientes. Neste contexto, o bar possui um (01) bartender que fica esperando os garçons na copa, possui *X* garçons, sendo que cada garçom consegue atender a um número limitado (C) de clientes por vez. Cada garçom somente vai para a copa para realizar o pedido quando todos os C clientes que ele pode atender tiverem feito o pedido, ou não houver mais clientes a serem atendidos. Um garçom deve entregar os pedidos aos seus clientes, e depois está liberado a atender os clientes novamente. Após ter seu pedido atendido, um cliente pode fazer um novo pedido após consumir seu pedido (o que leva um tempo aleatório). Por definição, nem todos os clientes precisam fazer um pedido a cada rodada. Implemente uma solução que permita a passagem por parâmetro de:

   a. A quantidade de clientes presentes no estabelecimento;
   b. A quantidade de garçons que estão trabalhando;
   c. A capacidade de atendimento dos garçons; e
   d. O número de rodadas que serão liberadas no bar.

Cada garçom e cada cliente devem ser representados por threads, estruturalmente definidos como os pseudocódigos que seguem (exemplos):

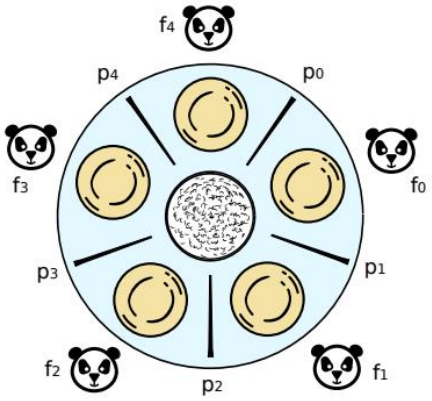
```
thread cliente {
    while (!fechouBar){
        fazPedido();
        esperaPedido();
        recebePedido();
        consomePedido(); //tempo variável
    }
}

thread garçom{
    while(existemClientesNoBar){
        recebeMaximoPedidos();
        registraPedidos();
        entregaPedidos();
        rodada++; //serve como parâmetro para fechar o bar
    }
}
```

A ordem de chegada dos pedidos dos clientes na fila de pedidos de cada garçom deve ser respeitada. Sua solução não deve permitir que clientes furem essa fila. O garçom só pode ir para a copa quando tiver recebido seus C pedidos. O programa deve mostrar a evolução, portanto planeje bem o que será apresentado. Deve ficar claro o que está acontecendo no bar a cada rodada: os pedidos dos clientes, os atendimentos pelos garçons, os deslocamentos para o pedido, a garantia de ordem de atendimento, etc.

02) **Barbeiro Dorminhoco**: Um barbeiro corta o cabelo de qualquer cliente. Se não há clientes, o barbeiro tira uma soneca. Há várias threads, uma para cada cliente. Um cliente aguarda pelo barbeiro se há ao menos uma cadeira vazia na barbearia, caso contrário, o cliente sai da barbearia imediatamente. Se há uma cadeira disponível, então o cliente senta. Se o barbeiro está dormindo, então o cliente acorda-o. Existem *N* cadeiras na barbearia. Faça um programa para a classe **BarbeiroDorminhoco** utlizando monitor.

03) **O Jantar dos Filósofos**: Neste problema, um grupo de cinco filósofos chineses alterna suas vidas entre meditar e comer. Ha uma mesa redonda com um lugar fixo para cada filósofo, com um prato, cinco palitos (hashis) compartilhados e um grande prato de arroz ao meio. Para comer, um filósofo f_i precisa pegar o palito à sua direita (p_i) e à sua esquerda (p_{i+1}), um de cada vez. Como os palitos são compartilhados, dois filósofos vizinhos não podem comer ao mesmo tempo. Os filósofos não conversam entre si nem podem

observar os estados uns dos outros. O problema do jantar dos filósofos é representativo de uma grande classe de problemas de sincronização entre vários processos e vários recursos sem usar um coordenador central. Resolver o problema do jantar dos filósofos consiste em encontrar uma forma de coordenar suas ações de maneira que todos os filósofos consigam meditar e comer.



04) Quanto aos clássicos problemas envolvendo *threads*:

a. Recrie o problema das Roletas para funcionar com qualquer quantidade de roletas
b. Simule diferentes ações em contas bancárias ao mesmo tempo (cada ação é uma thread)
    i. Transferência, depósito, crédito de juros
    ii. Saque, Depósito, crédito de juros, transferência
c. Crie um problema clássico de Produtor/Consumidor e resolva o mesmo utilizando semáforos e outro utilizando monitores