DI/CT/UFES - 2ª. Prova de Sistemas Operacionais I
Período: 2009/2 - Profa. Roberta Lima Gomes

Aluno: _____

1. **(2,0)** O que são *threads*? O que uma *thread* acrescenta ao modelo de processo? Quais os itens compartilhados por todas as *threads* de um processo? Diferencie *user-level threads* de *kernel-level threads*.

2. **(2,0)** Considere uma extensão do problema do produtor-consumidor definida da seguinte forma:
   a) Há dois processos consumidores e um processo produtor. Os processos executam concorrente e assincronamente.
   b) O produtor não pode escrever no buffer até que ambos os consumidores tenham lido dele. É aberta uma exceção para a primeira escrita.
   c) Nenhum consumidor pode ler o conteúdo de um buffer indefinido.
   d) Nenhum consumidor pode ler o mesmo conteúdo duas vezes sucessivamente.
   e) Ambos os consumidores podem executar seus eventos simultaneamente.

   Defina semáforos e use primitivas P e V para impor a exclusão mútua e sincronizar os processos.

```
Processo Produtor          Processo Consumidor1       Processo Consumidor2
...                        ...                        ...
repeat                     repeat                     repeat
  ...                        ...                        ...
  coloca dados no buffer     retira dados do buffer     retira dados do buffer
  ...                        ...                        ...
until forever              until forever              until forever
```

3. **(3,0)** Suponha que um grupo de N canibais come jantares a partir de uma grande travessa que comporta M porções. Quando alguém quer comer, ele(ela) se serve da travessa, a menos que ela esteja vazia. Se a travessa está vazia, o canibal acorda o cozinheiro e espera até que o cozinheiro coloque mais M porções na travessa. Desenvolva o código para as ações dos canibais e do cozinheiro. A solução deve evitar *deadlock* e deve acordar o cozinheiro apenas quando a travessa estiver vazia. Suponha um longo jantar, onde cada canibal continuamente se serve e come, sem se preocupar com as demais coisas na vida de um canibal...

4. **(3,0)** O código seguinte foi retirado do problema dos leitores e escritores:

```
GLOBAL VARIABLES: mutex, wrt : semaphores initialized to 1;
                  readcount  : integer initialized to 0;
READER: ......
       p(mutex);
       readcount := readcount+1;
       if readcount=1 then p(wrt);
       v(mutex)
       read
       p(mutex);
       readcount := readcount-1;
       if (readcount=0) then V(wrt);
       v(mutex);
       ....
WRITER:       .......
       p(wrt);
       write
       v(wrt);
       ....
```

   Modifique este programa de modo que o WRITER tenha precedência sobre o READER, isto é, se o WRITER estiver esperando e algum READER estiver lendo, nenhum outro READER poderá entrar na sua região crítica até que o WRITER tenha entrado e executado a sua respectiva RC.