MAC0438 – Programação Concorrente
Instituto de Matemática e Estatística
Aula de exercícios antes da P1 – 23 de Abril de 2013 **Quem entregar as soluções (ou tentativas) para 5 questões ganhará 0,75 na nota da P1**

Questão 1

Considere o seguinte programa:

```
int x = 0, y = 10;
co while (x != y) x = x + 1;
// while (x != y) y = y - 1;
oc
```

a) O programa atende aos requisitos da propriedade no máximo uma vez? Explique.

b) O programa termina sempre/algumas vezes/nunca? Explique.

Questão 2

Suponha que sua máquina tem a seguinte instrução atômica:

```
flip(lock)
   < lock = (lock + 1) % 2;
     return (lock); >
```

Um aluno de MAC0438 sugere a seguinte solução para o problema da seção crítica para dois processos:

```
int lock = 0;

process CS[i=1 to 2] {
   while (true) {
      while (flip(lock) != 1))
         while (lock != 0) skip;
      Secao critica;
      lock=0;
      Secao nao critica;
   }
}
```

a) Explique porque esta solução não atende a propriedade de exclusão mútua (Ou seja, mostre uma simulação detalhada em que dois processos estejam na seção crítica ao mesmo tempo).

b) Suponha que a primeira linha do `flip` seja modificada para fazer adições "módulo 3" ao invés de "módulo 2". Dessa forma a propriedade de exclusão mútua é atendida? Explique.

Questão 3

Suponha que um computador tenha a seguinte instrução atômica:

```
Swap (var1, var2):
   < tmp = var1; var1 = var2; var2 = tmp; >
```

Onde `tmp` é um registrador.

- Usando a instrução `Swap`, desenvolva uma solução para o problema da seção crítica para `n` processos. Não se preocupe com a propriedade de entrada garantida. Descreva claramente como sua solução funciona e atende as 3 outras propriedades.

Questão 4

Suponha que há `n` processos numerados de 1 a `n` e que um aluno de MAC0438 propôs o seguinte código para ser utilizado como uma barreira reutilizável:

```
int count = 0; go = 0;

codigo executado pelo Worker[1]:
   <await (count == n-1);>
   count = 0;
   go = 1;

codigo executado pelos Workers[2 a n]:
   <count++;>
   <await (go == 1);>
```

a) Explique o que há de errado no código.

b) Corrija o código. Não use mais nenhuma variável global. Você pode criar quantas variáveis locais forem necessárias. **O `< >` só pode ser usado para instruções de incremento ou com o `await`.**

Questão 5

A instrução `Fetch-and-Add`, `FA(var, incremento)`, é uma instrução atômica que retorna o valor antigo de `var` e adiciona `incremento` a esta variável. Usando a instrução `FA`, implemente as instruções `P` e `V` para um semáforo genérico `s`. Assuma que leituras e escritas em memória são atômicas mas que `FA` é a única instrução "poderosa".

Questão 6

Suponha que 1 processo produtor e `n` processos consumidores compartilham um buffer. O produtor deposita mensagens no buffer, consumidores leem essas mensagens. Cada mensagem depositada pelo produtor tem que ser lida por todos os `n` consumidores antes do produtor poder depositar outra mensagem no buffer.

a) Desenvolva uma solução para o problema. Utilize semáforos para sincronização entre processos.

b) Agora assuma que o buffer tem `b` posições. O produtor pode depositar mensagens apenas nos slots vazios ou reutilizáveis. Cada mensagem tem que ser lida por todos os `n` consumidores antes do slot referente àquela mensagem ser considerado como reutilizável. Além disso os consumidores precisam ler as mensagens na ordem que elas forem depositadas. Entretanto, diferentes consumidores podem ler mensagens em tempos diferentes. Por exemplo, em um instante de tempo um consumidor pode já ter recebido `b` mensagens a mais que outro consumidor mais lento. Desenvolva uma solução para esse novo problema. Utilize semáforos para sincronização entre processos.

Questão 7

Resolva o problema dos filósofos famintos focando no estado dos filósofos ao invés do estado dos garfos. Considere um vetor de booleanos `comendo[5]`. `comendo[i]` é verdade se o Filósofo `i` está comendo e falso caso contrário.

a) Desenvolva uma solução para o problema. Utilize semáforos para sincronização. Sua solução tem que ser livre de deadlock e não há problema se um filósofo morrer de fome (Dica: Use a técnica de passagem de bastão)

b) Modifique sua resposta em a) para evitar que algum filósofo morra de fome (Se a sua solução em a) já evita que algum filósofo morra de fome você não precisa fazer mais nada)