MAC0438 – Programação Concorrente
Instituto de Matemática e Estatística
Primeira Prova – 26 de abril de 2011

Nome: _____

Assinatura: _____

Nº USP: _____

Questão 1

Assumindo um escalonamento com justiça fraca, para quais valores iniciais de `x` o programa a seguir termina? Quais são os valores finais correspondentes? Explique sua resposta.

```
co < await (x >= 3) x = x - 3; >
// < await (x >= 2) x = x - 2; >
// < await (x == 1) x = x + 5; >
oc
```

Questão 2

No programa abaixo, as ações atômicas condicionais respeitam a propriedade no máximo uma vez? Justifique. Se as ações `<await (B)>` fossem substituídas por `while (!B)`, a variável compartilhada `x` estaria "protegida" de acessos simultâneos? Justifique.

```
int A = 0, B = 0, C = 1, x = 3, V1 = 1, V2 = 1;

process p {
   while (1) {
      A = 1; C = 1;
      < await (!B || C == 2); >
      if (x >= 3) x = x - 2;
      A = 0;
      V1 = V1 + 2;
   }
}

process q {
   while (1) {
      B = 1; C = 2;
      < await (!A || C == 1); >
      x++;
      B = 0;
      V2 = V2 + 2;
   }
}
```

Questão 3

Em 1987, Leslie Lamport propôs o algoritmo a seguir no famoso artigo *A Fast Mutual Exclusion Algorithm*. Avalie o algoritmo levando em conta as 4 propriedades necessárias para que ele resolva o problema da seção crítica. Justifique a avaliação de cada propriedade.

```
int x = 0, y = 0;

process p {
   while (1) {
      seção não crítica;
p1:   x = 1;
      if (y != 0) goto p1;
      y = 1;
      if (x != 1)
         if (y != 1) goto p1;
      seção crítica;
      y = 0;
   }
}

process q {
   while (1) {
      seção não crítica;
q1:   x = 2;
      if (y != 0) goto q1;
      y = 2;
      if (x != 2)
         if (y != 2) goto q1;
      seção crítica;
      y = 0;
   }
}
```

Questão 4

O comando `tee` do Unix é executado da seguinte forma:

```
tee arquivo
```

O `tee` lê linhas da entrada padrão e as escreve tanto na saída padrão quanto no arquivo `arquivo`. **(a)** Escreva um algoritmo do `tee` com 3 processos que rodem em paralelo. Um para ler da entrada padrão, um para escrever na saída padrão e um para escrever no arquivo `arquivo`. Use o estilo "`co` dentro de `while`". **(b)** Modifique o algoritmo de (a). Desta vez use o estilo "`while` dentro de `co`". Utilize quantos buffers forem necessários para que o programa possa ler e escrever em paralelo. Use `await` para sincronizar o acesso aos buffers.