Universidade Federal de Viçosa
Departamento de Informática
INF310 – Programação Concorrente e Distribuída
30/11/2018 – Prova 3 – Valor: 25 pontos

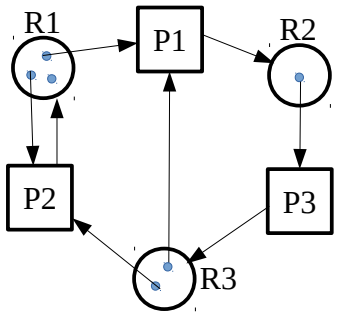
Nome: _____ Matrícula: _____

1. Responda as questões a seguir sobre sincronização de processos através de comunicação por troca de mensagens:
   a) **(3 pontos)** Quais as vantagens desta técnica sobre as vistas no início do curso como monitores ou semáforos?

   b) **(3 pontos)** Na internet, os roteadores precisam comunicar frequentemente para os roteadores vizinhos diretos (por exemplo, para saber quais roteadores estão ativos e descobrir roteadores recém-ligados). Que tipo de implementação para a primitiva de envio (*send*) deve ser utilizada com relação ao bloqueio e nomeação? Justifique.

2. Considere o grafo de alocação/requisição de recursos mostrado abaixo, para responder as questões:

   a) **(4 pontos)** O sistema encontra-se em deadlock? Justifique.



   b) **(3 pontos)** Suponha que mais uma instância de R1 seja alocada para P1. O sistema estaria em *deadlock* após esta alocação? Justifique.

3. Considere os processos (*tasks*) **esquerda** e **direita** mostrados a seguir.
   a) **(4 pontos)** Esboce o código da task **meio** – cabeçalho e corpo – detalhando os comandos de comunicação no estilo ADA.

```
task body esquerda is
  x : integer
  ...
  meio.recebe_esq(x)
  ...
```

```
task body direita is
  y : integer
  ...
  meio.recebe_dir(y)
  ...
```

   A implementação apresentada para a task **meio** deve atender aos seguintes requisitos:
   - permitir a recepção de um único valor de cada *task* (esquerda e direita);
   - a ordem de recepção deve ser qualquer;
   - com os valores recebidos a *task* **meio** deve calcular a função $y^2 + 2x + 8$;
   - o resultado deve ser enviado para outra *task* de nome **recebe** usando um ponto de entrada (*entry*) de nome **resultado** (não é necessário mostrar o código da *task* **recebe**).

   b) **(3 pontos)** Faça as alterações necessárias no corpo da *task* **meio** para permitir que as *tasks* esquerda e direita sejam transformadas em loops com um total de 10 interações entre elas.

4. **(5 pontos)** Considere o problema de um alocador de recursos. Existem 2 recursos R1 e R2 que precisam ser utilizados por processos de dois tipos A e B. A seguir são apresentados os códigos dos processos do tipo A e B.

```
process A(i: 1..N);
     ...
     send(controlador, 1);     //requisita R1
     receive(controlador, id1);
     send(controlador, 2);     //requisita R2
     receive(controlador, id2);
     “usando id1 e id2”;
     send(controlador, “fim”);
     ...
```

```
process B(i: 1..N);
     ...
     send(controlador, 2);     //requisita R2
     receive(controlador, id2);
     send(controlador, 1);     //requisita R1
     receive(controlador, id1);
     “usando id1 e id2”;
     send(controlador, “fim”);
     ...
```

Escreva o código do processo controlador de forma que não haja *deadlocks* e sem que um dos processos tenha prioridade sobre o outro. A comunicação entre os processos é síncrona. (Assuma que o controlador pode chamar uma função *getRID(R)* que retorna o identificador do recurso especificado por *R*. Por simplicidade, considere que existe apenas 1 unidade de cada recurso.)

Boa prova e boas férias!