Universidade de Brasília
Departamento de Ciência da Computação
Disciplina de Tópicos Avançados em Computadores - Programação Concorrente – Turma E
Prova 2 Período: 2014/2 Data: 26/11/2014
Aluno: Matrícula:

Questões

1 (1.5 pontos) - Explique o que são monitores, abordando como ocorre o controle de concorrência e a sincronização de processos em um monitor.

2 (1.5 pontos) - Explique como funciona a sincronização através de trocas de mensagens.

3 (2.5 pontos) - Complete o programa a seguir de maneira que o resultado impresso seja da forma "AAAAABBBBBCCC-CCDDDDD" (isto é, primeiro os A's depois os B's, seguidos pelos C's e por último os D's).

```
variáveis:
 . . .
 P_AC (int i)  /* processo i = 0,1,2,3,4, ..., n; executa o seguinte código */ {
       ...
       imprime('A');
       ...
       imprime('C');
        . . .
 }

 P_BD (int i)  /* processo i = 0,1,2,3,4, ..., m; executa o seguinte código */ {
       ...
       imprime('B');
       ...
       imprime('D');
        . . .
 }
```

4 (2.0 pontos) - Descreva como a solução abaixo para o jantar dos filósofos pode levar a um *deadlock*. Apresente ao menos uma alternativa para o problema de deadlock.

```
semaphore garfo[5] = 1,1,1,1,1; /* cada semáforo é inicializado com 1*/

philosopher (int i)  /* filósofo i=0,1,2,3,4 executa o seguinte código */     {
   while(1) {
      think();                      \\pensa
      sem_wait(garfo[i]);           \\pega um garfo
      sem_wait(garfo[(i+1)%5]);     \\pega o outro garfo
      eat();                        \\come
      sem_post(garfo[(i+1)%5]);     \\ devolve um garfo
      sem_post(garfo[i]);           \\devolve o outro garfo
   }
}
```

5 (2.5 pontos) - Implemente um semáforo usando um monitor. Chame este monitor de $monitor\_semaforo$ e implemente os métodos *up* e *down*.

```
monitor monitor_semaforo{

    variáveis:
        ...

     up() {
        ...
     }
     down(){
         ...
     }
}
```