	UNIVERSIDADE FEDERAL DA PARAÍBA CENTRO DE INFORMÁTICA	
	Disciplina	Linguagem de Programação II
	Semestre	2017.2
	Professor	Bruno Jefferson de Sousa Pessoa

Primeiro Exercício-Programa: Locks e Algoritmos Justos

1. O Problema da Montanha Russa

Suponha que existam *n* passageiros e um carro em uma montanha russa. Os passageiros, repetidamente, esperam para dar uma volta no carro. O carro tem capacidade para *C* passageiros, com $C < n$. O carro só pode partir quando estiver cheio. Após dar uma volta na montanha russa, cada passageiro passeia pelo parque de diversões e depois retorna à montanha russa para a próxima volta.

Tanto o carro quanto os passageiros devem ser representados por threads. A implementação das threads dos passageiros devem basear-se no seguinte pseudocódigo:

```
thread passageiro[i = 1 to n]  {
    while (!fechouParque) {
        entraNoCarro(); // Protocolo de entrada da solução justa

        /*  Incrementa  contador  que  registra  o  número  de
            passageiros transportados pelo carro.  */
        esperaVoltaAcabar();
        saiDoCarro(); // Protocolo de saída da solução justa
        passeiaPeloParque(); // tempo aleatório
    }
}
```

Da mesma forma, a implementação da thread do carro deverá a seguinte estrutura:

```
thread carro {
    while (existemPassageirosNoParque) {
        esperaEncher();
        daUmaVolta();
        esperaEsvaziar();
        volta++; // Indicador para o fechamento do parque.
    }
}
```

2. Instruções gerais

- O presente Exercício-Programa(EP) deve ser desenvolvido em grupo de no máximo 3 (três) alunos.
- O EP deve ser enviado por email até às 23:59 do dia **24/04/2017** e apresentado no dia seguinte.
- Deverá ser entregue um arquivo zipado, contendo o código fonte, no formato descrito a seguir:
    - LP2-EP1-grupo.zip
    - Ex.: LP2-EP1-jose-maria.zip

3. Instruções para implementação

- Este Exercício-Programa deve ser desenvolvido em Java ou em C++, utilizando a implementação de Threads pertencente ao seu núcleo.
- A implementação deverá atender às quatro propriedades de uma solução para o problema da seção crítica, a saber: Exclusão mútua, Ausência de *deadlock*, Ausência de atraso desnecessário e Entrada eventual.
- Para atender à propriedade da Entrada eventual, o programa deverá basear-se no **Algoritmo do Desempate para *n* Processos** apresentado em sala de aula.
- Instruções atômicas especiais, como *Test and Set* e *Fetch and Add,* devem ser utilizadas apenas em situações excepcionais, tais como a apresentação de texto na tela e operações aritméticas que não fazem parte da solução justa.
- O carro só pode partir se estiver cheio.
- No passeio pelo parque, as threads devem dormir um tempo aleatório dentro de um intervalo preestabelecido que tenha relação com o tempo da volta no carro, o qual deve ser constante.
- A saída do seu programa deve ser bem planejada, de forma a mostrar o que está acontecendo a cada momento, sem ficar carregada demais.
- A saída do programa deve apresentar, entre outros itens, as seguintes informações:
    - Identificador da volta atual realizada pelo carro;
    - Passageiro que entrou no carro;
    - Passageiro que saiu do carro;
    - Passageiro que está passando pelo parque;
    - O número de voltas dadas por cada passageiro.