Centro de Informática - UFPE
Segundo Exercício Escolar — IF686 (2017.2)
Data: 28/11/2017 Horário: 13h-15h
Docente: Márcio Lopes Cornélio
Discente:

Instruções para entrega: Enviar dois arquivos para *mlc2@cin.ufpe.br* : (1) arquivo ASCII com todas as respostas (copiar e colar os códigos das respostas) e (2) um arquivo no formato pdf com todas as respostas. Identificar-se nos arquivos (colocar nome). O assunto do email deve estar no formato: <login>_2ee_plc_2017_2

(2,0) 1. Uma conta bancária é compartilhada por seis pessoas. Cada uma pode depositar ou retirar dinheiro desde que o saldo não se torne negativo. Implemente em Java uma solução para conta bancária, utilizando bloco sincronizado. Considere que saques e depósitos são de valores aleatórios.

(2,0) 2. Adapte a solução anterior para utilizar tipos primitivos atômicos. Métodos *get* e *set* estão disponíveis para estes tipos.

(2,0) 3. Um semáforo binário é definido tradicionalmente pelas operações *p* e *v*. A primeira adquire um *lock*, fazendo o seguinte: testa se o *lock* é verdadeiro, se sim, adquire o *lock* tornando-o falso, senão espera. A segunda operação libera o *lock*, tornando-o verdadeiro. Defina um semáforo em Haskell usando STM. Para isso, defina um novo tipo com uma variável transacional do tipo *Bool*. Defina as funções *p* e *v*.

```
type Semaphore = TVar Bool


p :: Semaphore -> STM ()
p sem =  do {     b <- readTVar sem ;
                  if b
                  then writeTVar sem False
                  else retry
          }

v :: Semaphore -> STM ()
v sem = writeTVar sem True
```

(2,0) 4. Implemente a classe Semáforo em Java com dois métodos que realizam as seguintes operações

- up - incrementa o contador do semáforo e acorda threads que estejam bloqueadas
- down - decrementa o contador do semáforo ou, caso seja zero, suspende a thread

```
class Semaforo extends Object
{
private int count;

public Semaforo(int startingCount){
count = startingCount;
}

public void down(){
synchronized (this) {
while (count <= 0) {
```

```
// Aguarda
try {
wait();
  } catch (InterruptedException ex) {
  }
 }
 count--;
 }
}

public void up(){
synchronized (this) {
count++;
//notifica uma thread para acordar
if (count == 1 ) {
notify();
   }
  }
 }
}
```

(2,0) 5. Na preparação de sanduíches em uma lanchonete, uma pessoa fornece os ingredientes (pão, carne e tomate); duas outras são responsáveis por preparar os sanduíches. Porém, a lanchonete dispõe de apenas uma faca para ser utilizada na preparação. Considere que os recipientes de ingredientes são continuamente reabastecidos na capacidade máxima de porções (30 para cada ingrediente). Desenvolva uma solução em Haskell que modele o funcionamento desta lanchonete. Utilize memória transacional e variáveis mutáveis.