CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS
CAMPUS VII - UNIDADE TIMÓTEO - Engenharia da Computação

Laboratório de Programação Concorrente
Prof. Lucas Pantuza Amorim

Barreiras e *pool* de *threads*

O objetivo deste Laboratório é praticar a implementação de sincronização por barreira e *pool* de *threads*. Para cada atividade, siga o roteiro proposto e responda a todas as questões colocadas antes de passar para a próxima atividade.

Atividade 1

Objetivo: Avaliar e completar uma implementação de um *pool* de *threads* em Java.

Roteiro:

1. Edite o arquivo `MyPool.java` e compreenda o que faz e como funciona.
2. Documente a classe `FilaTarefas`.
3. Execute o programa (várias vezes) alterando o tamanho do *pool* de *threads* e o número de tarefas executadas. Avalie se o programa funciona como esperado.
4. Acrescente um novo tipo de tarefa para ser executada no mesmo *pool* de *threads*: dado um número inteiro positivo verifica se ele é primo (o método que verifica se determinado número é primo está disponível abaixo). Complete a implementação da classe `Primo` e descomente as linhas 95 e 96 do programa.

```
1  //função para determinar se um numero é primo
2  int ehPrimo(long unsigned int n) {
3      int i;
4          if(n<=1) return 0;
5          if(n==2) return 1;
6          if(n%2==0) return 0;
7          for(i=3; i<sqrt(n)+1; i+=2) {
8              if(n%i==0) return 0;
9          }
10         return 1;
11 }
```

5. Teste sua implementação.

Relatório da atividade: Em um arquivo `.txt` descreva as respostas das questões colocadas e inclua exemplos dos resultados obtidos nas execuções realizadas.

Atividade 2

Objetivo: Usar semáforos ou variáveis de condição para implementar uma barreira.

Roteiro:

1. Implemente uma aplicação com N *threads*, onde cada *thread* executa a seguinte função:

```c
#define NITER 5
int vet[N]; //variavel global

void *thread (void *threadid) {
  int *tid = (int*) threadid; aux=1;
  vet[*tid] = *tid;
  printf("Thread %d chegou na barreira %d\n", *tid, aux);
  barreira(); //implementar espera coletiva
  printf("Thread %d saiu da barreira %d\n", *tid, aux);
  aux++;

  for (int i=0; i<NITER; i++) {
     vet[*tid] = vet[*tid] * vet[*tid] + vet[*tid-1];
     printf("Thread %d chegou na barreira %d\n", *tid, aux);
     barreira(); //implementar espera coletiva
     printf("Thread %d saiu da barreira %d\n", *tid, aux);
     aux++;
  }
  free(tid);
}
```

2. Implemente a função `barreira()`.
3. Execute o programa várias vezes e avalie os resultados.

Relatório da atividade: Em um arquivo `.txt` descreva as respostas das questões colocadas e inclua exemplos dos resultados obtidos nas execuções realizadas.