INE 5645 – PROGRAMAÇÃO PARALELA E DISTRIBUIDA – PROVA 2 – 12/06/2017

ALUNO _____

1. **Sockets - Indicar (Verdade/Falso):**
   (a) (Verdade/Falso) *Sockets* são abstrações utilizadas nos protocolos de comunicação UDP e TCP, que implementam pontos de interação para comunicação entre processos.
   (b) (Verdade/Falso) *Sockets* só podem ser usados em Linux.
   (c) (Verdade/Falso) A comunicação entre processos consiste em transmitir uma mensagem entre o *socket* de um processo e o socket de outro processo.
   (d) (Verdade/Falso) Para que um processo receba mensagens, seu *socket* deve estar vinculado ao IP de um computador e a uma porta local desse computador em que é executado.
   (e) (Verdade/Falso) As mensagens enviadas para um endereço IP e uma porta específica, só podem ser recebidas por um processo cujo *socket* esteja associada a esse IP e a esse número de porta.
   (f) (Verdade/Falso) Processos podem usar o mesmo *socket* para enviar e receber mensagens.
   (g) (Verdade/Falso) Qualquer processo pode fazer uso de várias portas para receber mensagens.
   (h) (Verdade/Falso) Um processo não pode compartilhar portas com outros processos no mesmo computador.
   (i) (Verdade/Falso) Os processos que usam *IP Multicast* são uma exceção, pois esses compartilham portas.
   (j) (Verdade/Falso) Qualquer número de processos podem enviar mensagens para a mesma porta.

2. **(Formas de Escalonamento em Java)**

   No código seguinte,

```
1. ExecutorService executar_leitor Executors.newFixedThreadPool(4);
2. ScheduledExecutorService executar_escritor =
                                Executors.newScheduledThreadPool(1);

      ..............................
      try
      {
3. executar_leitor(new Leitor( sharedLocation ));
4. executar_escritor.scheduleAtFixedRate(new
                 Escritor(sharedLocation),0,1,TimeUnit.MILLISECONDS)
      }
      ..................................
```

   (a) Qual a diferença entre `ExecutorService e ScheduledExecutorService`?

(b) O que se pode afirmar sobre o **estado das threads** Leitor e Escritor, quando o trecho da linha 4 for executado ?

(c) (Verdade/Falso) No código da linha 1, as threads no pool são escalonadas em paralelo.

(d) (Verdade/Falso) No código da linha 2, as threads no pool são escalonadas concorrentemente.

3. **Controle de Concorrência – Semáforo**

Considere a seguinte a definição de semáforo binário S. Seja o pseudo-código:

```
S: Semaphore := 1;

Thread T1 is begin
loop
   Non_Critical_Section;
   wait(S);
   Critical_Section_1;
   signal(S);
   Non_Critical_Section;
end loop;
End T1;

Thread T2 is begin
loop
   Non_Critical_Section;
   wait(S);
   Critical_Section_2;
   signal(S);
   Non_Critical_Section;
end loop;
End T2;
```

Considerando níveis de prioridade de execução de 1 a 10, a Thread T1 é de menor prioridade e Thread T2, a de maior prioridade. Indique a resposta (E) errada e (C) correta.

( ) A Thread T1 entra no processador e executa até seu final. Em seguida a Thread T2 é executada.

( ) A Thread T1 entra no processador, ela é interrompida, e em seguida Thread T2 é executada e interrompida antes de chegar ao seu final.

( ) Se a Thread T1 entra no processador, ela é interrompida, e em seguida Thread T2 é escalonada e executada até ao seu final.

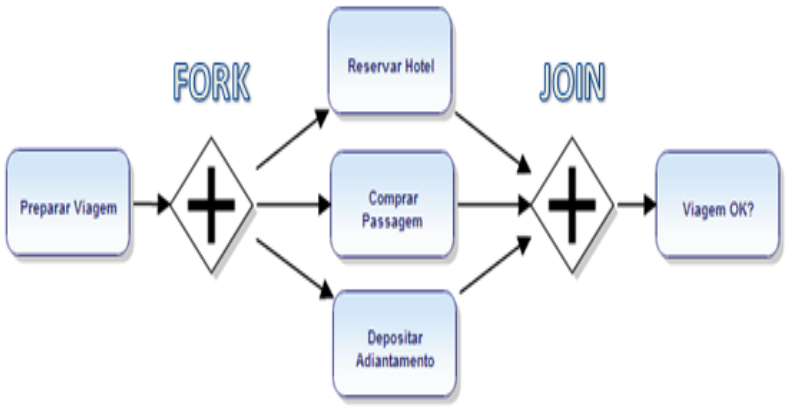
( ) O esquema de escalonamento fundamental para threads é preemptivo (o ato de forçar uma thread parar sua execução), ou seja, baseado em prioridade.
( ) *Starvation* ocorrerá neste cenário.

4. Componentes de processos e threads

(Verdade/Falso) Itens propriedade de **threads** são: (a) Espaço de endereçamento, (b) Variáveis globais, (c) Contador de programa lógico, (d) Registradores, (e) Pilha, (f) Estado, (g) Recursos.

(Verdade/Falso) Itens propriedade de **processos** são: (a) Variáveis globais, (b) Contador de programa lógico, (c) Registradores, (d) Pilha, (e) Estado.

5. OpenMP (Modelo de Programação)



(a) No exemplo acima, como se denomina o modelo de programação do OpenMP ?

(b) (Verdade/Falso) – A região paralela indicada representa uma região paralela contendo seções paralelas.

6. OpenMp -

(a) Sobre o código seguinte pode afirmar (SIM/NÃO)

```
#define N 10000;
int i;
#pragma omp parallel
   #pragma omp for
        for (i=0 ; i < 10000 ; i++) {
            calculo();
        }
        printf("Terminado");
```

(1) As iterações são distribuídas entre as threads ?

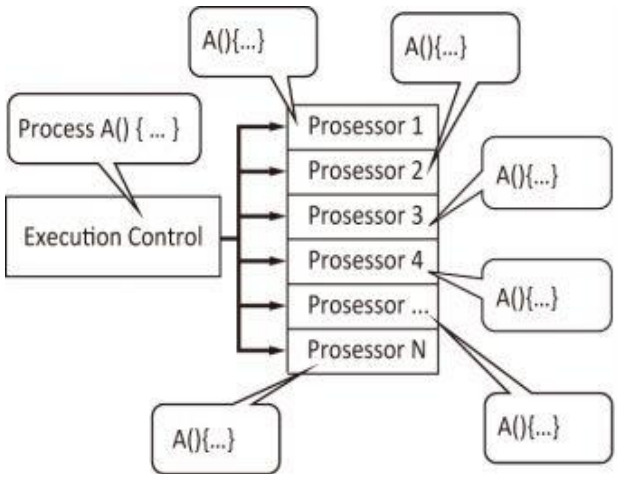
(2) Tem uma barreira implícita de sincronização entre threads no final do loop ?

(3*) omp for* pode ser complementado pela schedule para especificar como fazer a distribuição da carga do *for* (i=0 ; i < 10000 ; i++) ?

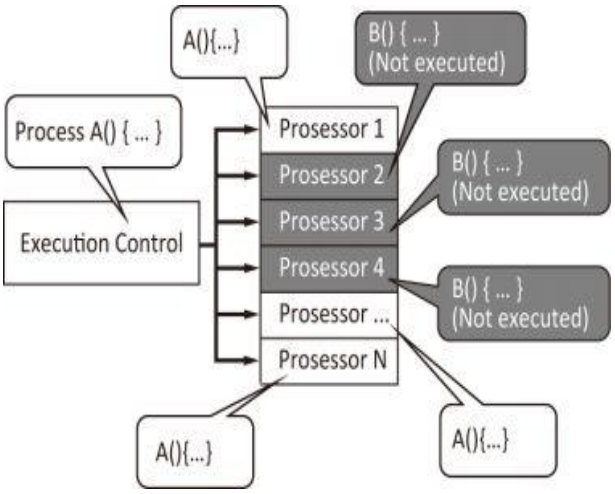
(b) Vimos que o OpenMP irá particionar automaticamente as iterações de um loop *for*. Como podemos otimizar a forma como as iterações de loop são divididas. No código abaixo, indique no pontilhado, qual tipo de schedule é mais apropriado ?

```
#define THREADS 4
#define N 16
int main ( ) {
 int i;

 #pragma omp parallel for schedule(.........) num_threads(THREADS)
 for (i = 0; i < N; i++) {
   /* wait for i seconds */
   sleep(i);

   printf("Thread %d has completed iteration %d.\n",
                                          omp_get_thread_num( ), i);
  }
  /* all threads done */
  printf("All done!\n");
  return 0;
}
```

7. OpenCL – Dadas as figuras abaixo, o que significam cada uma delas ?



Resposta:_____
_____
_____
_____



Resposta:_____
_____
_____
_____

8. **OpenCL – Indique quais afirmações estão erradas e as que estão corretas.**

( ) (**Paralelismo de Tarefas**) Uma técnica de programação que divide uma grande quantidade de dados em partes menores que podem ser operadas em paralelo. A mesma operação é executada simultaneamente (isto é, em paralelo).

( ) (**Paralelismo de Dados**) Uma técnica de programação que divide uma grande quantidade de dados em partes menores que podem ser operadas em paralelo. A mesma operação é executada simultaneamente (isto é, em paralelo).

( ) As **arquiteturas** apropriadas para *Paralelismo de dados* é "SIMD" e para *Paralelismo de tarefas* é "MIMD".

( ) **SIMD** significa que as unidades paralelas têm instruções distintas, então cada uma delas pode fazer algo diferente em um dado momento.

( ) **SIMD** significa que todas as unidades paralelas compartilham a mesma instrução, mas a realizam em diferentes elementos de dados.

( ) No **modelo de paralelismo de tarefas**, para cada operação a ser executada, deve ser definido um kernel para executar uma determinada operação na arquitetura MIMD.

9. **OpenCL -** Dê um exemplo, envolvendo uma operação sobre números, que seja executada com paralelismo de dados.

10. OpenCL

(a) No código abaixo, escrito em C, **quantos núcleos** são utilizados, se um processador quad-core for utilizado no processamento ?

```
void ArrayDiff(const int* a, const int* b, int* c, int n)
{
  for (int i = 0; i < n; ++i)
  {
    c[i] = a[i] - b[i];
  }
}
```

Resposta: _____

(b) Um objeto de programa encapsula o *código-fonte de um kernel*, sendo este identificados no código-fonte por meio da palavra-chave __kernel. O que significa, o seguinte código, executado em OpenCL ?

```
__kernel void ArrayDiff (
   __global const int* a,
    __global const int* b,
    __global int* c )
{
  int id = get_global_id(0);
  c[id] = a[id] - b[id];
}
```