



[Caio Martins Ferreira](#)

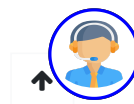
Iniciado em segunda-feira, 17 jul. 2023, 14:07

Estado Finalizada

Concluída em segunda-feira, 17 jul. 2023, 15:57

Tempo empregado 1 hora 50 minutos

Avaliar **2,20** de um máximo de 10,00(**22%**)



Questão 1

Completo

Atingiu 0,10 de 1,50

No código a seguir, os pragmas declarados nas linhas 11 e 14 garantem a divisão equilibrada do trabalho entre o total de threads especificadas na variável de ambiente OMP_NUM_THREADS.

```
1  #include <stdio.h>
2  #include <omp.h>
3  #define TAM 12
4  int main () {
5      int A[TAM], B[TAM], C[TAM];
6      int i;
7      for (i=0; i<TAM; i++) {
8          A[i]=2*i - 1;
9          B[i]= i + 2;
10     }
11     #pragma omp parallel
12     {
13         int tid = omp_get_thread_num();
14         #pragma omp for
15         for (i=0; i<TAM; i++) {
16             C[i] = A[i] + B[i];
17             printf("Thread[%d] calculou C[%d]\n", tid, i);
18         } /* fim-for */
19     } /* fim-pragma */
20     for (i=0; i<TAM; i++)
21         printf("C[%d]=%d\n", i, C[i]);
22 } /* fim-main */
23
```

Apresente uma nova versão desse código que garanta a distribuição equilibrada de trabalho entre as threads (de acordo com o valor de OMP_NUM_THREADS), considerando apenas o pragma de paralelização descrito na linha 11 (ou seja, assuma a não existência do pragma da linha 14).

```
include <stdio.h>
```

```
#include <omp.h>
```

```
#define TAM 12
```

```
int main() {
```

```
    int A[TAM], B[TAM], C[TAM];
```

```
    int i;
```

```
    for(i=0; i<TAM; i++) {
```

```
        A[i] = 2*i -1;
```

```
        B[i] = i +2;
```

```
    }
```

```
    #pragma omp parallel private(i)
```

```
    {
```

```
        int tid = omp_get_thread_num();
```

```
        i = TAM/omp_get_num_threads();
```

```
        int n_casa = tid+i;
```

```
        i *= tid;
```

```
        for(i; i<n_casa; i++) {
```

```
            C[i] = A[i] +B[i];
```

```
        }
```



```

}
for(i=0; i<TAM; i++){
    printf("C[%d]=%d\n", i, C[i]);
}
return 0;
}

```

⚙️ [01.c](#)

Comentário:

Histórico de respostas

Passo	Hora	Ação	Estado	Pontos
1	17/07/23, 14:07:35	Iniciada	Ainda não respondida	
2	17/07/23, 15:46:30	Salvou: include <stdio.h> #include <omp.h> #define TAM 12 int main() { int A[TAM], B[TAM], C[TAM]; int i; for(i=0; i<TAM; i++) { A[i] = 2*i -1; B[i] = i +2; } # pragma omp parallel private(i) { int tid = omp_get_thread_num(); i = TAM/omp_get_num_threads(); int n_casa = tid+i; i *= tid; for(i; i<n_casa; i++) { C[i] = A[i] +B[i]; } } for(i=0; i<TAM; i++) { printf("C[%d]=%d\n", i, C[i]); } return 0; } Anexos: {\$ a}	Resposta salva	
3	17/07/23, 15:57:46	Tentativa finalizada	Completo	
4	28/07/23, 16:57:37	Avaliado manualmente com 0.1 e comentário: Fernando William Cruz	Completo	0,10



Questão 2

Incorreto

Atingiu 0,00 de 0,75

O código a seguir foi compilado com OpenMP e o binário tem o nome t1. Com base no código fonte, analise as afirmações e marque V para as verdadeiras e F para as falsas.

```
1  #include <stdio.h>
2  #include <omp.h>
3
4  int main(int argc, char *argv[]) {
5
6      int i=0;
7      #pragma omp parallel
8      {
9          if (omp_get_thread_num() == 1)
10             i=i+10;
11     }
12     printf("i=%d\n", i);
13     return 0;
14 }
```

I - A execução com o comando **OMP_NUM_THREADS=4 t1** vai imprimir o valor 40

II - Se a linha 9 for suprimida, o binário equivalente acionado com o comando **OMP_NUM_THREADS=3 t1** imprimirá sempre o valor 30

III - Se na linha 7 for acrescentada a declaração **private(i)** e houver supressão da linha 9, o binário equivalente acionado com o comando **OMP_NUM_THREADS=6 t1**, o programa vai imprimir 60


- ☐ a. Apenas as afirmativas I e II estão corretas
- ☐ b. Apenas as afirmativas I e III estão corretas
- ☐ c. Nenhuma das opções apresentadas corresponde às afirmativas apresentadas
- ☐ d. Apenas a afirmativa I está correta
- ☒ e. Apenas as afirmativas II e III estão corretas ✖

Sua resposta está incorreta.

A resposta correta é:

Nenhuma das opções apresentadas corresponde às afirmativas apresentadas

Histórico de respostas

Passo	Hora	Ação	 Estado	Pontos
<u>1</u>	17/07/23, 14:07:35	Iniciada	Ainda não respondida	
<u>2</u>	17/07/23, 15:20:13	Salvou: Apenas as afirmativas II e III estão corretas	Resposta salva	
3	17/07/23, 15:57:46	Tentativa finalizada	Incorreto	0,00

Questão 3

Incorreto

Atingiu 0,00 de 0,75

Analise as afirmações a seguir e marque a alternativa correta

I - O problema dos generais bizantinos é uma metáfora que descreve a dificuldade de se entrar em um acordo quando entidades centralizadas decidem em nome da maioria

II - A tecnologia blockchain é uma solução eficiente para o problema dos generais bizantinos

III - O algoritmo Paxos é uma solução de consenso distribuído cuja variante pode ser usada para coordenação e resolução de impasses em redes blockchain

- ☐ a. Apenas II e III estão corretas
- ☐ b. Apenas II está correta
- ☐ c. Apenas I e III estão corretas
- ☒ d. Apenas I e II estão corretas ✖
- ☐ e. Todas estão corretas

Sua resposta está incorreta.

A resposta correta é:

Apenas II e III estão corretas

Histórico de respostas

Passo	Hora	Ação	Estado	Pontos
<u>1</u>	17/07/23, 14:07:35	Iniciada	Ainda não respondida	
<u>2</u>	17/07/23, 14:32:08	Salvou: Apenas I e II estão corretas	Resposta salva	
3	17/07/23, 15:57:46	Tentativa finalizada	Incorreto	0,00

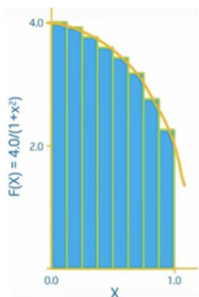


Questão 4

Completo

Atingiu 0,20 de 1,50

O programa a seguir é uma implementação do cálculo da função Pi em modo serial, usando único espaço de endereçamento. Com base neste código crie uma versão MPI, de modo a permitir que os processos, em conjunto e colaborativamente, consigam encontrar o valor de Pi, pela distribuição do cálculo das áreas dos retângulos entre os processos.



Matematicamente, sabemos que:

$$\int_0^1 \frac{4.0}{1+x^2} dx = \pi$$

Podemos aproximar essa integral como a soma de retângulos:

$$\sum_{i=0}^n F(x_i) \Delta x \cong \pi$$

Onde cada retângulo tem largura Δx e altura $F(x_i)$ no meio do intervalo i .

```
1  #include <stdio.h>
2  #define NUM_STEPS 8000000
3
4  int main(void) {
5      double x, pi, sum=0.0;
6      double step;
7
8      step = 1.0/(double) NUM_STEPS;
9      for (int i=0; i<NUM_STEPS; i++) {
10         x = (i+0.5) * step;
11         sum+=4/(1.0+x*x);
12     } /*fim-for */
13     pi = sum*step;
14     printf("Pi = %f\n", pi);
15 } /*fim-main */
```

```
#include <stdio.h>
```

```
#include <mpi.h>
```

```
#define TAG 0
```

```
#define MESTRE 0
```

```
#define NUM_STEPS 8000000
```

```
int main() {
```

```
int rank, size;
```

```
double resp = 0.0;
```

```
double temp = 0.0;
```

```
double x, pi, sum = 0.0;
```

```
MPI_Init(&argc, &argv);
```

```
MPI_Comm_rank(MPI_COMM_WORLD, &rank);
```

```
MPI_Comm_size(MPI_COMM_WORLD, &size);
```

```
int pt = (int)NUM_STEPS/size
```

```
double step = 1.0 / (double)NUM_STEPS;
```

```
if(rank == 0) {
```

```
for(int i=1; i<size; i++){
```

```
MPI_Send(&pt, 1, MPI_INT, i, TAG, MPI_COMM_WORLD); //(mensagem, quantidade, type, destino(rank), TAG, grupoMPI)
```

```
}
```

```
for(int i=1; i<size; i++){
```

```
MPI_Recv(&temp, 1, MPI_INT, i, TAG, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
```

```
resp += temp;
```

```
}
```

```
} else {
```

```
MPI_Recv(&valor, 1, MPI_INT, MESTRE, TAG, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
```

```
for(int i; i<NUM_STEPS; i++){
```

```
x = (i+0.5) * step;
```

```
sum += 4 / (1.0+x*x);
```

```
}
```



```

resp = valor * 2;
MPI_Send(&resp, 1, MPI_INT, MESTRE, TAG, MPI_COMM_WORLD);
printf("asdfasdf");
}

```

```

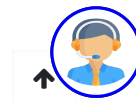
pi = sum*step;
printf("pi = %f\n", pi);
}

```

Comentário:

Histórico de respostas

Passo	Hora	Ação	Estado	Pontos
<u>1</u>	17/07/23, 14:07:35	Iniciada	Ainda não respondida	
<u>2</u>	17/07/23, 15:56:50	Salvou: #include <stdio.h> #include <mpi.h> #define TAG 0 #define MESTRE 0 #define NUM_STEPS 8000000 int main() { int rank, size; double resp = 0.0; double temp = 0.0; double x, pi, sum = 0.0; MPI_Init(&argc, &argv); MPI_Comm_rank(MPI_COMM_WORLD, &rank); MPI_Comm_size(MPI_COMM_WORLD, &size); int pt = (int)NUM_STEPS/size double step = 1.0 / (double)NUM_STEPS; if(rank == 0) { for(int i=1; i<size; i++){ MPI_Send(&pt, 1, MPI_INT, i, TAG, MPI_COMM_WORLD); //(mensagem, quantidade, type, destino(rank), TAG, grupoMPI) } for(int i=1; i<size; i++){ MPI_Recv(&temp, 1, MPI_INT, i, TAG, MPI_COMM_WORLD, MPI_STATUS_IGNORE); resp += temp; } } else { MPI_Recv(&valor, 1, MPI_INT, MESTRE, TAG, MPI_COMM_WORLD, MPI_STATUS_IGNORE); for(int i; i<NUM_STEPS; i++){ x = (i+0.5) * step; sum += 4 / (1.0+x*x); } resp = valor * 2; MPI_Send(&resp, 1, MPI_INT, MESTRE, TAG, MPI_COMM_WORLD); printf("asdfasdf"); } pi = sum*step; printf("pi = %f\n", pi); }	Resposta salva	
<u>3</u>	17/07/23, 15:57:46	Tentativa finalizada	Completo	
4	28/07/23, 20:27:43	Avaliado manualmente com 0.2 e comentário: Fernando William Cruz	Completo	0,20



Questão 5

Correto

Atingiu 0,75 de 0,75

Analise o código a seguir e responda o que se segue

```
1  #include <stdio.h>
2  #include <omp.h>
3  int main(){
4      int tid=0, nthreads=0;
5      printf("\nRegião serial (thread única)\n\n");
6      #pragma omp parallel
7      {
8          tid = omp_get_thread_num();
9          nthreads = omp_get_num_threads();
10         printf("Região paralela (thread %d de %d threads)\n", tid, nthreads);
11     } /*fim-pragma */
12     printf("\nRegião serial (thread única)\n\n");
13     #pragma omp parallel num_threads(4)
14     {
15         tid = omp_get_thread_num();
16         nthreads = omp_get_num_threads();
17         printf("Região paralela (thread %d de %d threads)\n", tid, nthreads);
18     } /* fim-pragma */
19     printf("\nRegião serial (thread única)\n\n");
20     return 0;
21 } /* fim-main */
```

1. Se OMP_NUM_THREADS=6, na segunda região paralela desse código (linhas 13 a 18), serão geradas 10 threads e, portanto, 10 impressões (linha 17)
2. Se a linha 15 for movida para ficar fora da região paralela (entre as linhas 11 e 13), esse código passa a ser não compilável, pois não é possível saber o número de threads em uma região serial do código
3. Esse código é mais apropriado para funcionar em arquiteturas UMA (Uniform Memory Access) ou de memória compartilhada do que em arquiteturas NUMA (Non Uniform Memory Access)

- ☐ a. Nenhuma das alternativas apresentadas é válida
- ☐ b. Apenas a primeira e a terceira afirmação está correta
- ☐ c. Apenas a segunda e a terceira afirmação está correta
- ☐ d. Apenas a primeira afirmação está correta
- ☒ e. Apenas a terceira afirmação está correta ✓

Sua resposta está correta.

A resposta correta é:

Apenas a terceira afirmação está correta



Histórico de respostas

Passo	Hora	Ação	Estado	Pontos
<u>1</u>	17/07/23, 14:07:35	Iniciada	Ainda não respondida	
<u>2</u>	17/07/23, 15:14:01	Salvou: Apenas a terceira afirmação está correta	Resposta salva	
3	17/07/23, 15:57:46	Tentativa finalizada	Correto	0,75

Questão 6

Incorreto

Atingiu 0,00 de 0,75

O código a seguir foi compilado com OpenMP e o binário tem o nome t1. Com base no código fonte, analise as afirmações e marque V para as verdadeiras e F para as falsas.

```
1  #include <stdio.h>
2  #include <omp.h>
3
4  int main(int argc, char *argv[]) {
5
6      int i=0;
7      #pragma omp parallel
8      {
9          if (omp_get_thread_num() == 1)
10             i=i+10;
11     }
12     printf("i=%d\n", i);
13     return 0;
14 }
```

I - A execução com o comando **OMP_NUM_THREADS=1 t1** vai imprimir o valor zero para a variável i

II - Se na linha 7 for acrescentada a declaração **private(i)**, o binário equivalente acionado com o comando **OMP_NUM_THREADS= 2 t1** vai imprimir o valor 20

III - Ao suprimir a linha 9, o binário equivalente vai imprimir valores aleatórios para a variável i, desde que o número de threads seja maior que 1

- ☐ a. Apenas a afirmativa I está correta
- ☒ b. Nenhuma das opções apresentadas consegue julgar corretamente as afirmativas ✖
- ☐ c. Apenas as afirmativas II e III estão corretas
- ☐ d. Apenas as afirmativas I e II estão corretas
- ☐ e. Apenas as afirmativas I e III estão corretas

Sua resposta está incorreta.

A resposta correta é:

Apenas as afirmativas I e III estão corretas

Histórico de respostas



Passo	Hora	Ação	Estado	Pontos
<u>1</u>	17/07/23, 14:07:35	Iniciada	Ainda não respondida	
<u>2</u>	17/07/23, 15:22:58	Salvou: Nenhuma das opções apresentadas consegue julgar corretamente as afirmativas	Resposta salva	
3	17/07/23, 15:57:46	Tentativa finalizada	Incorreto	0,00

Questão 7

Completo

Atingiu 0,00 de 1,50

Elabore um programa OpenMP que imprima os elementos de uma matriz $M[100:8]$ posições (do tipo int), considerando o seguinte:

- Apenas uma das threads deve inicializar a matriz, com a fórmula: $M[i][j]=i+j$
- A matriz deve ser impressa de modo colaborativo por todas as threads ativadas, da linha 0 até a linha 99, nesta ordem
- Cada thread deve imprimir pelo menos uma das linhas da matriz
- Cada thread, uma vez acionada, deve imprimir a matriz a partir da linha que ainda não foi impressa
- Considerar que este programa pode ser executado por, no máximo, 6 threads
- O número de posições a serem impressas deve obedecer a um **offset** dinâmico, ou seja, um valor randômico – menor que 15 – que é calculado por cada thread, no momento em que a thread é escalonada para imprimir a matriz
- O programa deve controlar a impressão de modo que a matriz inteira seja impressa em ordem e de modo que nenhuma posição seja impressa mais de uma vez. Por exemplo, se a matriz já foi impressa até a linha 18 e o **offset** dinâmico calculado foi 10, a thread atual deve imprimir da linha 19 considerando 10 posições adiante
- O programa termina quando a matriz M inteira, linha por linha, tiver sido impressa.

```
#include <stdio.h>

#include <omp.h>

#define TAM_I 100
#define TAM_J 8

int main() {
    int M[TAM_I][TAM_J];
    int i, j;

    for(i=0; i<TAM_I; i++) {
        for(j=0; j<TAM_J; j++) {
            M[i][j] = i+j;
        }
    }

    #pragma omp parallel private(pos)
    {
        int tid = omp_get_thread_num();
        pos = TAM_I*TAM_J/omp_get_num_threads();
        int n_casa = tid+i;
        pos *= tid;
        for(i; i<n_casa; i++) {
            C[i] = A[i] +B[i];
        }
    }

    for(i=0; i<TAM; i++) {
        printf("C[%d]=%d\n", i, C[i]);
    }

    return 0;
}
```



Comentário:

Histórico de respostas

Passo	Hora	Ação	Estado	Pontos
<u>1</u>	17/07/23, 14:07:35	Iniciada	Ainda não respondida	
<u>2</u>	17/07/23, 15:57:39	Salvou: <pre>#include <stdio.h> #include <omp.h> #define TAM_I 100 #define TAM_J 8 int main() { int M[TAM_I][TAM_J]; int i,j; for(i=0; i<TAM_I; i++){ for(j=0; j<TAM_J; j++){ M[i][j] = i+j; } } # pragma omp parallel private(pos) { int tid = omp_get_thread_num(); pos = TAM_I*TAM_J/omp_get_num_threads(); int n_casa = tid+i; pos *= tid; for(i; i<n_casa; i++){ C[i] = A[i] +B[i]; } } for(i=0; i<TAM; i++){ printf("C[%d]=%d\n", i, C[i]); } return 0; }</pre>	Resposta salva	
<u>3</u>	17/07/23, 15:57:46	Tentativa finalizada	Completo	
4	28/07/23, 11:06:26	Avaliado manualmente com 0 e comentário: Fernando William Cruz	Completo	0,00



Questão 8

Incorreto

Atingiu 0,00 de 0,75

O código a seguir foi compilado com OpenMP e o binário tem o nome t1. Com base no código fonte, analise as afirmações e marque V para as verdadeiras e F para as falsas.

```
1  include <stdio.h>
2  #include <omp.h>
3  #include <string.h>
4  #define MAX 100
5
6  int main(int argc, char *argv[]) {
7
8      #pragma omp parallel
9      {
10         int soma=0;
11         #pragma omp for
12         for (int i=0;i<MAX;i++) {
13             soma +=omp_get_num_threads()*i;
14         } /* fim-for */
15         printf("Thread[%d] iterou %d vezes\n",
16                omp_get_thread_num(), soma);
17     } /* fim omp parallel */
18     return 0;
19 }
```

I - A execução com o comando **OMP_NUM_THREADS=4 t1** vai imprimir que cada thread foi executada 25 vezes

II - Se este programa for acionado tendo a variável OMP_NUM_THREADS um valor maior do que o número de núcleos da máquina, apenas as threads equivalentes ao número de núcleos serão criadas

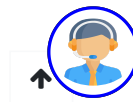
III - Se o programa for executado numa máquina com 10 núcleos de processamento e a variável OMP_NUM_THREADS estiver com valor igual a 20, o programa não será ativado

- ☐ a. Apenas as afirmativas II e III estão corretas
- ☐ b. Apenas a afirmativa II está correta
- ☐ c. Apenas a afirmativa III está correta
- ☒ d. Apenas a afirmativa I está correta ✖
- ☐ e. Todas as afirmativas estão erradas

Sua resposta está incorreta.

A resposta correta é:

Todas as afirmativas estão erradas



Histórico de respostas

Passo	Hora	Ação	Estado	Pontos
<u>1</u>	17/07/23, 14:07:35	Iniciada	Ainda não respondida	
<u>2</u>	17/07/23, 14:21:08	Salvou: Apenas a afirmativa I está correta	Resposta salva	
3	17/07/23, 15:57:46	Tentativa finalizada	Incorreto	0,00

Questão 9

Completo

Atingiu 0,40 de 1,00

Por quê os algoritmos de hash consistente são necessários em redes P2P? Que tipo de problemas esse algoritmo resolve? Na resposta, apresentar um exemplo esclarecendo o funcionamento do algoritmo.

um tabelamento em hash em uma rede P2P permite a adição e exclusão de nós na rede com maior facilidade. Ele ajuda na manutenção da listagem de endereços.

Comentário:

Histórico de respostas

Passo	Hora	Ação	Estado	Pontos
<u>1</u>	17/07/23, 14:07:35	Iniciada	Ainda não respondida	
<u>2</u>	17/07/23, 14:48:51	Salvou: um tabelamento em hash em uma rede P2P permite a adição e exclusão de nós na rede com maior facilidade. Ele ajuda na manutenção da listagem de endereços.	Resposta salva	
<u>3</u>	17/07/23, 15:57:46	Tentativa finalizada	Completo	
4	28/07/23, 20:50:57	Avaliado manualmente com 0.4 e comentário: Fernando William Cruz	Completo	0,40



Questão 10

Correto

Atingiu 0,75 de 0,75

O código a seguir foi compilado com OpenMP e o binário tem o nome t1. Com base no código fonte, analise as afirmações e marque V para as verdadeiras e F para as falsas.

```
1  #include <stdio.h>
2  #include <omp.h>
3
4  int main(void) {
5      printf(" %d\n", omp_get_num_threads());
6      #pragma omp parallel
7      {
8
9      }
10     printf("%d\n", omp_get_max_threads());
11     return 0;
12 }
```

I - A execução com o comando **OMP_NUM_THREADS=4 t1** vai imprimir o valor 4 na linha 5 e o valor 12 na linha 10, se o computador onde esse programa estiver rodando tiver 12 núcleos

II - Este programa vai imprimir sempre o valor 1 na linha 5, independente do número de threads definidas na variável OMP_NUM_THREADS

III - O comando da linha 10 vai imprimir sempre o valor 1, uma vez que este está fora da região paralela definida pelo **pragma omp parallel**


- ☒ a. Nenhuma das opções consegue julgar as afirmativas apresentadas ✓
- ☐ b. Apenas as afirmativas I e II estão corretas
- ☐ c. Apenas as afirmativas I e III estão corretas
- ☐ d. Apenas as afirmativas II e III estão corretas
- ☐ e. Apenas a afirmativa I está correta

Sua resposta está correta.

A resposta correta é:

Nenhuma das opções consegue julgar as afirmativas apresentadas

Histórico de respostas

Passo	Hora	Ação		Estado	Pontos
<u>1</u>	17/07/23, 14:07:35	Iniciada		Ainda não respondida	
<u>2</u>	17/07/23, 15:55:10	Salvou: Nenhuma das opções consegue julgar as afirmativas apresentadas		Resposta salva	
3	17/07/23, 15:57:46	Tentativa finalizada		Correto	0,75