```c
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <time.h>

/*|Matheus Martins Batista (2019005687) - Sistemas Operacionais (COM120) |
  |Ciências da Computação (CCO) - EP03 - Exercício01 - 03/10/2021        |*/
#define NUM_THREADS 2

int maior = 0, menor = 0, n;
float medio = 0;

void *maior_numero(void *arg){ //passar o argumento(vetor) para função comparar o elementos e selecionar o maior
    int i;
    int *numeros = (int*) arg;
    for(i=0;i<n;i++){
        if(numeros[i]>maior){
            maior = numeros[i];
        }
    }
    return NULL;
}

void *menor_numero(void *arg){ //passar o argumento(vetor) para função comparar o elementos e selecionar o menor
    int i;
    int *numeros = (int*) arg;
    menor = numeros[0];
    for(i=0;i<n;i++){
        if(numeros[i]<menor){
            menor = numeros[i];
        }
    }
    return NULL;
}

void *calcula_media(void *arg){ //passar o argumento(vetor) para função calcular a média
    int i;
    float soma = 0;
    int *numeros = (int*) arg;
    for(i=0;i<n;i++){
        soma = soma + numeros[i];
    }
    medio = soma/n;
    return NULL;
}

int main(int argc, char *argv[]){
    n = atoi(argv[1]); //converter o argumento para inteiro
    int *numeros = (int*)malloc(n*sizeof(int)), i;
    srand(time(NULL)); //semente para gerar os números aleatórios

    if(numeros){ //verificar alocação
        for(i=0;i<n;i++){
            numeros[i] = rand() % 100; //números aleatórios 0 a 100
            //printf("%d\n", numeros[i]); descomentar para conferir os elementos da lista
        }
    }
    else{
        printf("Falha ao alocar memória!");
```

```c
        return -1;
    }

    pthread_t threads[NUM_THREADS];
    //Thread 0 calcula o valor médio
    pthread_create(&threads[0], NULL, calcula_media, (void*) numeros);
    pthread_join(threads[0], NULL);

    //Thread 1 busca o maior valor
    pthread_create(&threads[1], NULL, maior_numero, (void*) numeros);
    pthread_join(threads[1], NULL);

    //Thread 2 busca o menor valor
    pthread_create(&threads[2], NULL, menor_numero, (void*) numeros);
    pthread_join(threads[2], NULL);

    printf("A média dos valores é: %.4f\n", medio);
    printf("O maior valor é: %d\n", maior);
    printf("O menor valor é: %d\n", menor);
    free(numeros); //liberar o vetor
    return 0;
}
/* Compilado com gcc -Wall ex01.c -lpthread -o ./ex01, o argumento é passado pela linha de comando
do terminal.O programa gera os valores com rand() e usando como semente a função time(), são criadas
e juntadas as threads de forma linear, cada uma faz a sua operação, ao fim do processo as variáveis
globais são acessadas na main para expor os resultados, não houve problemas de sincronismo durante
a execução utilizando o pthread_join().*/
```