Nome:_____

Nº ESTG: _____

Aval. Periódica - EI/Prog. Avançada

Teste Prático – 2022.10.29/A

Licenciatura em
Engenharia Informática
UC de Programação Avançada
2º ano – Engenharia Informática
Regime diurno/pós-laboral
Ano letivo 2022/2023 - 1ºSemestre

Teste Prático – Enunciado A

2022.10.29 / 09h30' Prova com consulta **Duração:** 80 minutos

Nome Completo:_____

N.º de Estudante:_____ **Regime:** [ ] Diurno [ ] Pós-laboral

IMPORTANTE

É expressamente proibido o recurso à Internet durante a prova. Qualquer utilização não autorizada da Internet leva à anulação da prova e ao reportar da situação às autoridades competentes. O mesmo sucede com outros tipos de tentativa de fraude.

- **Antes de iniciar a prova:**
  - Execute os seguintes comandos:

    `cd; mkdir -p ~/ProvaP/R_NUMERO/`

    (em que **R** deve ser substituído pela letra **D** se for do regime diurno e **N** se for aluno do regime pós-laboral e **NUMERO** deve ser substituído pelo seu número ESTG);
  - Para garantir que o seu diretório de trabalho seja o correto, faça:

    `cd ~/ProvaP/R_NUMERO/`
- **Após ter terminado a prova:**
  - Deverá proceder à criação de um arquivo TAR, fazendo uso do seguinte comando:

    `cd ~/ProvaP/R_NUMERO/; tar cvf ProvaP_YYYYMMDD_R_NUMERO.tar *`

    (em que YYYYMMDD corresponde à data corrente, e.g., 20221029, e R_NUMERO obedece ao formato acima indicado);
  - Verifique que o arquivo ".tar" que criou não está vazio, através da execução de:

    `tar tvf ProvaP_YYYYMMDD_R_NUMERO.tar`
  - Entregue o arquivo ".tar" através da plataforma moodle, no espaço reservado para o efeito. Em caso de dúvidas, pergunte ao professor;
  - Informe o professor para este validar a receção dos seus ficheiros.

Pergunta [20 valores]

(Escreva as suas respostas a esta pergunta no diretório **`"~/ProvaP/R_NUMERO/Pergunta"`**. Deve indicar o seu nome completo e número de estudante Politécnico de Leiria no ficheiro **README.txt** a ser criado no diretório)
NOTA 1: não é permitida a chamada a comandos externos através da função *system* ou de outra com funcionalidade similar.
NOTA 2: a solução deve ser implementada com recurso aos ficheiros do diretório **EmptyProject-Template.zip**
NOTA 3: código entregue que **não compile** através do utilitário *make* e do respetivo *makefile* na máquina virtual da UC leva à atribuição da classificação de **0 (zero) valores** à resposta.

Recorrendo à linguagem C, implemente o programa **concurrent_matrix** que exemplifica o acesso concorrente de vários *threads* a uma *matriz* partilhada. O programa deverá criar um processo filho, sendo que o processo pai deverá, no final da execução do programa, escrever na saída padrão "**#END#**", garantindo que essa é a última mensagem do programa. Por sua vez, o processo filho deve criar **number_threads** *threads*. Cada *thread* deverá escrever **number_writes** vezes numa matriz de 5 linhas por 4 colunas do tipo `int`. No total deverão ocorrer **number_threads x number_writes** operações de escrita na matriz. Cada *thread* deverá escolher uma posição aleatório da matriz e em seguida colocar um valor aleatório inteiro entre 1 e 1000

nessa posição, repetindo este processo **number_writes** vezes. O programa deverá permitir que uma *thread* A escreva na posição **i,j** da matriz, e simultaneamente uma outra *thread* B escreva na posição **x,y**, desde que i,j seja distinto de x,y, ou seja, nesse caso o *thread* A não deve esperar pelo *thread* B e vice-versa.

Antes de terminar, o programa deverá escrever no canal de saída padrão o número de escritas realizadas em cada um dos elementos da matriz, assim como o número total de escritas efetivamente realizadas.

A aplicação **concurrent_matrix** deverá aceitar o seguinte parâmetro da linha de comandos:

-t/--number_threads <int>: número de threads a criar. Caso não seja especificado, deve ser considerado o valor 100. O valor deve ser maior ou igual a 1. Parâmetro opcional.

-w/--number_writes <int>: número de operações de escrita a realizar na matriz. Caso não seja especificado, deve ser considerado o valor 1024. O valor deve ser maior ou igual a 1. Parâmetro opcional.

Nota: Poderá usar a função *rand_r* para gerar números aleatórios diferentes para cada *thread* como é exemplificado no código abaixo.

```
void *thread_func(void *arg) {
       unsigned int rand_state = time(NULL) ^ getpid() ^ pthread_self();
       int value1 = rand_r(&rand_state);      // gera um valor aleatório
       int value2 = rand_r(&rand_state);      // gera outro valor aleatório
       return NULL;
}
```

Considere os seguintes exemplos.

Exemplo nº1

```
$./concurrent_matrix -t 50 -w 100

Number of writes for each element:
[0,0]: 237
[0,1]: 238
[0,2]: 239
[0,3]: 271
[1,0]: 260
[1,1]: 259
[1,2]: 265
[1,3]: 254
[2,0]: 272
[2,1]: 253
[2,2]: 243
[2,3]: 238
[3,0]: 250
[3,1]: 240
[3,2]: 256
[3,3]: 241
[4,0]: 246
[4,1]: 246
[4,2]: 239
[4,3]: 253
Total number of writes: 5000

#END#
```

Exemplo nº2

```
$./concurrent_matrix -t 1 -w 1

Number of writes for each element:
[0,0]: 0
[0,1]: 0
[0,2]: 0
[0,3]: 0
[1,0]: 0
[1,1]: 0
[1,2]: 1
[1,3]: 0
[2,0]: 0
[2,1]: 0
[2,2]: 0
[2,3]: 0
[3,0]: 0
[3,1]: 0
[3,2]: 0
[3,3]: 0
[4,0]: 0
[4,1]: 0
[4,2]: 0
[4,3]: 0
Total number of writes: 1

#END#
```

Exemplo nº3

```
$./concurrent_matrix -t -1 -w 1000

number_threads must be a positive integer.
```

Exemplo nº4

```
#
$./concurrent_matrix -t 100 -w -1

number_writes must be a positive integer.
```