Nome:_____

Nº ESTG: _____

Aval. Periódica - EI/Prog. Avançada

Prova prática 2 – 2017.12.16/C

Licenciatura em
Engenharia Informática
UC de Programação Avançada
2º ano – Engenharia Informática
Regime diurno e pós-laboral
Ano letivo 2017/2018 - 1ºSemestre

2º Teste Prático – Enunciado C

2017.12.16 / 09h30 | Prova com consulta | **Duração:** 90 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 ~/Prova02/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 ~/Prova02/R_NUMERO/
    ```

- **Após ter terminado a prova:**
  - Deverá proceder à criação de um arquivo TAR, fazendo uso do seguinte comando:

    ```
    cd ~/Prova02/R_NUMERO/;  tar cvf Prova02_YYYYMMDD_R_NUMERO.tar *
    ```

    (em que YYYYMMDD corresponde à data corrente (e.g., 20171216) e R_NUMERO obedece ao formato acima indicado);
  - Verifique que o arquivo ".tar" que criou não está vazio, através da execução de:

    ```
    tar tvf Prova02_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 **"~/Prova02/R_NUMERO/Pergunta"**. Deve indicar o seu nome completo e número de estudante IPLeiria 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_Client-Server-Template_v1.x** do arquivo **templates_PA.zip**.

NOTA 3: código entregue que **não compile** através do utilitário *make* e do respetivo *makefile* leva à atribuição da classificação de **0 (zero) valores** à resposta.

NOTA 4: assuma que não existe perda de datagramas entre os programas cliente e servidor.

(continua na próxima página)

Pretende-se que implemente, recorrendo à linguagem C, uma aplicação cliente/servidor que permita efetuar sequências de lançamentos de moedas. Os programas cliente e servidor devem comunicar através do protocolo de transporte UDP.

A aplicação servidora **head_tail_srv** é uma aplicação UDP iterativa que deve gerar um número aletório (0 ou 1), de modo a simular o lançamento de uma moeda, resultando em *Tail* (Coroa) ou *Head* (Cara), respetivamente. Esta aplicação recebe os seguintes argumentos de entrada:

- **-p, --port <INT>**: porto UDP **[opção obrigatória]**
- **-s, --seed <INT>**: número inteiro empregue para inicializar o gerador de números aleatórios **[opção não obrigatória]**

A opção "**-s/--seed**" é empregue para inicializar o gerador de números aleatórios com um número inteiro, que deve estar entre 0 e 2^{16}-1. Caso a opção "**-s/--seed**" não seja especificada (é uma opção não obrigatória), o gerador de números aleatórios deve ser inicializado com o valor 1.

A aplicação servidora **head_tail_srv** deve esperar por um datagrama com o número de lançamentos de moeda a realizar e responder com um inteiro sem sinal de **32 *bits*** representando os resultados obtidos de forma ordenada. Cada um dos *bits* deste inteiro corresponde a um lançamento de moeda, sendo que 0 representa *Tail* e 1 representa *Head*. Concretamente, o *bit* menos significativo codifica o resultado do primeiro lançamento, o segundo *bit* menos significativo codifica o resultado do segundo lançamento e assim sucessivamente. O servidor não deverá realizar sequências de lançamentos **inferiores a 1 ou superiores a 16**. Caso um pedido da aplicação cliente solicite uma sequência fora destes limites, o servidor deverá apresentar uma apropriada mensagem de erro no canal de erro padrão, enviando à aplicação cliente o valor de 32 bits **com todos os *bits* a 1**.

A aplicação cliente **head_tail_clnt** recebe os seguintes argumentos de entrada através da linha de comando:

- **-p, --port <INT>**: porto UDP **[opção obrigatória]**
- **-i, --ip <IPv4>**: endereço IPV4 (formato dotted-decimal) da aplicação servidor a contactar **[opção obrigatória]**
- **-f, --flips <INT>**: número inteiro empregue para solicitar lançamentos de moeda **[opção obrigatória]**

A opção "**-f/--flips**" representa o número de lançamentos a solicitar ao servidor e que deve estar entre 0 e 255.

Todos os parâmetros passados à aplicação servidora e à aplicação cliente devem ser convenientemente validados.

A aplicação cliente deve começar por enviar um datagrama com o número de lançamentos de moeda a realizar. Caso o pedido tenha sucesso, esta deverá mostrar por ordem os resultados de cada um dos lançamentos simulados na saída padrão. Em caso de erro, deverá ser apresentada uma mensagem de erro apropriada no canal de erro padrão.

Considere os seguintes exemplos que ilustram o funcionamento das aplicações.

Exemplo #1
("semente" não especificada através da linha de comando):

```
Servidor:
        $ ./head_tail_srv -p 8899
        Server: UDP/8899
        Seed: 1
        Waiting for coin flip requests...
        Flipping coin 3 times.
        Heads -> 2
        Tails -> 1
        Waiting for coin flip requests...

Cliente:
        $ ./head_tail_clnt -i 127.0.0.1 -p 8899 -f 3
        Requesting 3 coin flips...
        Results:
        [01] Head
        [02] Tail
        [03] Head
```

Exemplo #2
(número de lançamentos não suportado pelo servidor):

```
Servidor:
        $ ./head_tail_srv -p 8899
        Server: UDP/8899
        Seed: 1
        Waiting for coin flip requests...
        Unsuported flip number (17 times).
        Waiting for coin flip requests...

Cliente:
        $ ./head_tail_clnt -i 127.0.0.1 -p 8899 -f 17
        Requesting 17 coin flips...
        The requested number of flips is not supported by
        the server (17).
```

Exemplo #3
("semente" especificada através da linha de comando):

```
Servidor:
        $ ./head_tail_srv -p 8899 -s 1000
        Server: UDP/8899
        Seed: 1000
        Waiting for coin flip requests...
        Flipping coin 10 times.
        Heads -> 5
        Tails -> 5
        Waiting for coin flip requests...
Cliente:
        $ ./head_tail_clnt -i 127.0.0.1 -p 8899 -f 10
        Requesting 10 coin flips...
        Results:
        [01] Tail
        [02] Head
        [03] Head
        [04] Tail
        [05] Tail
        [06] Head
        [07] Head
        [08] Tail
        [09] Tail
        [10] Head
```