 <b>ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO</b>	Tipo de Prova Mini Teste 2 (Época de Recurso)	Ano letivo 2019/2020	Data 14/02/2020
	Curso Licenciatura em Engenharia Informática Licenciatura em Segurança Informática em Redes de Computadores	Hora 10:00	
	Unidade Curricular Fundamentos de Programação	Duração 2:00 horas	

Observações:

- Pode trocar a ordem das questões, desde que as identifique convenientemente.
- É permitida a utilização da biblioteca de input de dados (API\_Leitura.h) abordada nas aulas, disponibilizada em anexo.
- Todos os exercícios devem ser resolvidos utilizando a linguagem de programação C.
- Qualquer tentativa de fraude implica a anulação do exame.


Na resolução desta prova considere as estruturas de dados que são apresentadas de seguida, que servem de suporte a uma consola que permite jogar ao conhecido Jogo do Galo. Para um jogador jogar tem obrigatoriamente que registar-se, dando o seu nome e idade, sendo-lhe atribuído um identificador numérico único. A consola guarda todos os jogos, sendo guardado um apontador para cada um dos dois jogadores, o dia, mês e ano em que decorreu o jogo, o nº de jogadas e o estado final do tabuleiro. Por último, guarda-se ainda para cada jogo a lista de jogadas e o seu tamanho. Cada jogada tem a linha e coluna (um inteiro entre 0 e 2) bem como a letra jogada.

```
#define MAX_STR 100
#define MAX_JOGADAS 9

struct jogador{
    unsigned int id;
    char nome[MAX_STR];
    unsigned short idade;
};

struct jogada{
    unsigned short linha, coluna;
    char letra;
};

struct jogo{
    struct jogador *player1, *player2;
    unsigned short dia, mes, ano, n_jogadas;
    struct jogada jogadas[MAX_JOGADAS];
    char tabuleiro[3][3];
};
```

 <b>ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO</b>	Tipo de Prova Mini Teste 2 (Época de Recurso)	Ano letivo 2019/2020	Data 14/02/2020
	Curso Licenciatura em Engenharia Informática Licenciatura em Segurança Informática em Redes de Computadores		Hora 10:00
	Unidade Curricular Fundamentos de Programação		Duração 2:00 horas

1. Implemente a função **preenche\_tabuleiro**, cuja assinatura é apresentada abaixo. Dados 2 apontadores para dois jogadores **pl1** e **pl2**, esta função pesquisa o primeiro jogo entre os dois jogadores na lista de jogos e preenche o tabuleiro desse jogo de acordo com a lista de jogadas recebidas como argumento. A função devolve -1 se não existir qualquer jogo entre os dois jogadores ou se houver coordenadas inválidas na lista de jogadas. Caso contrário, devolve o nº de jogadas colocadas no tabuleiro.

(2.5 V)

(15 min)


```
/**
 * Função que pesquisa, num array de jogos, o primeiro jogo entre os jogadores
 * pl1 e pl2 e preenche o tabuleiro desse jogo de acordo com as jogadas
 * recebidas como parâmetro. A função devolve -1 se o jogo não existir ou se
 * alguma jogada tiver coordenadas inválidas, e devolve um número >= 0 que
 * representa o número de jogadas colocadas no tabuleiro.
 *
 * @param pl1 apontador para o primeiro jogador a pesquisar
 * @param pl2 apontador para o segundo jogador a pesquisar
 * @param jogos array de jogos
 * @param n_jogos tamanho do array de jogos
 * @param jogadas array de jogadas a copiar para o tabuleiro do jogo
 * @param n_jogadas tamanho do array de jogadas
 * @return -1 se o jogo não existir ou se coordenadas inválidas, ou nº de
 * jogadas colocadas no tabuleiro
 */
int preencheTabuleiro(struct jogador* pl1, struct jogador* pl2,
    struct jogo* jogos, unsigned short n_jogos,
    struct jogada* jogadas, unsigned short n_jogadas);
```

2. Implemente a função **filtra\_jogos**, cuja assinatura é apresentada abaixo. Esta função copia para uma nova lista (**jogos\_pl1**) todos os jogos da lista de jogos em que o jogador **pl1** tenha participado. A função deve alocar a nova lista utilizando apenas o espaço estritamente necessário, preenchendo ainda a variável respetiva ao seu tamanho final (**n\_jogos\_pl1**). A função devolve 1 em caso de sucesso, ou 0 caso contrário

(4V)

(30 min)

```
/**
 * Função que copia, para uma nova lista, os jogos em que um determinado
 * jogador já participou. A função deve alocar o espaço estritamente
 * necessário para os dados, e preencher tanto a lista como o seu tamanho.
 *
 * @param pl1 apontador para o jogador a pesquisar
 * @param jogos array de jogos
 * @param n_jogos tamanho do array de jogos
 * @param jogos_pl1 apontador duplo para o array a preencher com os jogos em
 * que o jogador pl1 já participou
 * @param n_jogos_pl1 apontador para o tamanho do array a preencher
 * @return 1 em caso de sucesso, 0 caso contrário
 */
int filtraJogos(struct jogador* pl1, struct jogo* jogos, unsigned short n_jogos,
    struct jogo** jogos_pl1, unsigned short *n_jogos_pl1);
```

 ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO	Tipo de Prova Mini Teste 2 (Época de Recurso)	Ano letivo 2019/2020	Data 14/02/2020
	Curso Licenciatura em Engenharia Informática Licenciatura em Segurança Informática em Redes de Computadores		Hora 10:00
	Unidade Curricular Fundamentos de Programação		Duração 2:00 horas

3. Implemente a função **registar\_jogador**, cuja assinatura é apresentada abaixo. Esta função adiciona um novo jogador, com a informação recebida como argumento, à lista de jogadores. Se a lista estiver cheia a função deve acrescentar 5 novas posições antes de inserir o novo jogador. A função deve devolver 1 em caso de sucesso ou 0 caso contrário.


(3V)  
(20 min)

```
/**
 * Função que acrescenta um novo jogador à lista de jogadores, caso não
 * exista já um jogador com o mesmo id. Se a lista estiver cheia a função
 * deve alocar 5 novas posições antes de inserir a informação do novo
 * jogador. A função retorna 1 em caso de sucesso e 0 caso contrário.
 */
* @param jogadores apontador duplo para a lista de jogadores
* @param n_jogadores número de elementos na lista de jogadores
* @param max_jogadores tamanho da lista de jogadores
* @param id identificador do jogador a acrescentar
* @param nome nome do jogador a acrescentar
* @param idade idade do jogador a acrescentar
* @return 1 em caso de sucesso, 0 caso contrário
*/
int registaJogador(struct jogador **jogadores, unsigned int *n_jogadores,
                  unsigned int *max_jogadores, unsigned int id, char *nome,
                  unsigned short idade);
```

4. Implemente a função **get\_jogador\_mais\_novo**, cuja assinatura é apresentada abaixo. Esta função pesquisa pelo jogador mais novo (com menor idade) que tenha realizado um jogo. Na situação em que existam mais que um jogador com a menor idade, considere o primeiro jogador encontrado. A função deve retornar um apontador para o jogador encontrado em caso de sucesso, ou NULL caso contrário.

(3.5V)  
(20min)

```
/**
 * Função que procura pelo jogador mais novo que tenha participado num jogo.
 * Após a sua identificação, deverá retornar um apontador para o mesmo. A função
 * retorna um apontador para o jogador mais novo em caso de sucesso, NULL caso
 * contrário.
 */
* @param jogos array de jogos
* @param n_jogos tamanho do array de jogos
* @param jogadores lista de jogadores
* @param n_jogadores número de elementos na lista de jogadores
* @return Apontador para o jogador mais novo em caso de sucesso, NULL caso
*         contrário
*/
struct jogador * getJogadorMaisNovo(struct jogo* jogos, unsigned short n_jogos,
                                     struct jogador *jogadores, unsigned int n_jogadores);
```

 <small>ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO</small>	Tipo de Prova Mini Teste 2 (Época de Recurso)	Ano letivo 2019/2020	Data 14/02/2020
	Curso Licenciatura em Engenharia Informática Licenciatura em Segurança Informática em Redes de Computadores	Hora 10:00	
	Unidade Curricular Fundamentos de Programação	Duração 2:00 horas	

5.  
(3.5V)  
(15 min)


Implemente a função **verifica\_jogada\_inicial**, cuja assinatura é apresentada abaixo. Esta função conta o número total de jogos em que uma posição (dada como argumento) foi utilizada como jogada inicial. Pode considerar apenas a posição (linha e coluna), e ignorar a letra que foi colocada nessa posição. A função deve retornar o número total de vezes em que a jogada foi utilizada como jogada inicial.

```
/**
 * Função que verifica quantas vezes uma jogada foi utilizada como jogada
 * inicial.
 *
 * @param jogos array de jogos
 * @param n_jogos tamanho do array de jogos
 * @param jogada_inicial jogada utilizada para comparação
 * @return total de vezes em que a jogada foi utilizada como primeira jogada
 * do jogo.
 */
int verificaJogadaInicial(struct jogo* jogos, unsigned short n_jogos,
    struct jogada jogada_inicial);
```

6.  
(3.5V)  
(20 min)

Considere agora que se pretende alterar o funcionamento da consola do Jogo do Galo da seguinte forma:

- É necessário registar o resultado do jogo, isto é, saber quem ganhou/perdeu ou se terminou com um empate (1V);
- É necessário registar o número de vitórias, empates e derrotas de cada jogador bem como todos os jogos em que este já participou (1.5V);
- É necessário registar o número de cada jogada (1V);

 <b>ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO</b>	Tipo de Prova Mini Teste 2 (Época de Recurso)	Ano letivo 2019/2020	Data 14/02/2020
	Curso Licenciatura em Engenharia Informática Licenciatura em Segurança Informática em Redes de Computadores	Hora 10:00	
	Unidade Curricular Fundamentos de Programação	Duração 2:00 horas	

## Anexo

Conteúdo do header file API\_Leitura.h:

```
void readShort(short *const value, const short minValue,
               const short maxValue, char const* const message);

void readInt(int *const value, const int minValue,
             const int maxValue, char const* const message);

void readLong(long *const value, const long minValue,
              const long maxValue, char const* const message);

void readFloat(float *const value, const float minValue,
               const float maxValue, char const* const message);

void readDouble(double *const value, const double minValue,
                const double maxValue, char const* const message);

void readChar(char *const value, char const* const message);

bool readString(char *const value, const unsigned int size,
                char const* const message);

void readBool(bool *const value, char const* const message);
```

Conteúdo parcial do header file string.h:

```
size_t strlen (const char *s)
char * strcpy (char *restrict to, const char *restrict from)
int strcmp (const char *s1, const char *s2)
```