Prova com consulta. Duração: 70 min. Parte prática [15/20 valores]

O exame é preenchido e submetido através do Moodle.

1. [4 valores] Tendo por base a biblioteca de vetores **`vetor.c/.h`**, implemente as seguintes alíneas. Sempre que conveniente utilize as funções disponíveis.

1.1. Faça a função **`vetor_novo()`** que cria um vetor novo vazio. **Dica**: Não se esqueça de inicializar os diferentes componentes do novo vetor.

1.2. Crie a seguinte função de teste que cria um vetor e insere as três strings fornecidas na posição final do vetor:

```
vetor *vetor_teste(char *s1, char *s2, char *s3)
```

2. [6 valores] A loja ELECOMP é uma loja especializada na venda de componentes eletrónicos. Para tal, precisam de uma base de dados para gerir o stock no armazém dos componentes de eletrónica e as compras dos clientes.

Considere os ficheiros disponibilizados:

- **`componentes.h`**: declarações das funções da biblioteca do armazém, bem como dos registos envolvidos, nomeadamente **`armazem`**, **`componente`**, **`fatura`** e **`parcela`**.
- **`componentes.c`**: ficheiro onde deverão ser implementadas as funções pedidas, relativas à biblioteca.
- **`armazem-teste.c`**: inclui o programa principal que invoca e realiza testes básicos às funções implementadas.
- **`db_small.txt`**: ficheiro de texto com informações sobre os componentes existentes no armazém.

Tendo por base o armazém de componentes, implemente uma nova função, **`parcela *parcela_remove(fatura *f, const char *ID)`**, que tem por objetivo remover uma parcela da fatura (parâmetro **`f`**) dado o seu ID (parâmetro **`ID`**). A função deve retornar o apontador para a parcela removida da fatura ou `NULL` se ocorrer um erro ou a parcela não existir na fatura. Sempre que conveniente utilize as funções disponíveis na biblioteca fornecida.

Verifique que:

- No registo **`parcela`** são guardados: 1) a identificação (`ID`) do componente; 2) um inteiro com a quantidade vendida (`quantidade`); 3) um *float* com o preço do componente no momento da compra (`preco`); 4) um apontador para o próximo elemento (`proximo`). Este registo contém a informação de cada parcela de uma fatura.

```
typedef struct _parcela
{
    /** ID do componente **/
    char ID[10];

    /** Quantidade de componentes comprados **/
    int quantidade;

    /** Preco unitário do componente no momento da compra **/
    float preco;

    /** Apontador para a proxima parcela **/
    struct _parcela *proximo;

} parcela;
```

- O registo de dados **`fatura`** aponta para a primeira parcela da fatura. Inclui também o número de parcelas da fatura (`n_parcelas`) e o preço total da fatura (`preco_total`). Este registo guarda a informação para se poder fazer a fatura de uma compra.

```
typedef struct
{
    /* Apontador para a primeira parcela */
    parcela *inicio;

    /* Numero de parcelas da fatura*/
    int n_parcelas;

    /* Preco total da fatura */
    float preco_total;
} fatura;
```

3. [5 valores] Considere novamente a biblioteca **componentes.h**.

Adicionalmente, foi criado o registo de dados **compra** que caracteriza a recolha em armazém das parcelas de uma fatura. **compra** representa uma pilha cujos elementos são do tipo **parcela**, organizados por ordem crescente de quantidade, ou seja, no topo da pilha encontra-se sempre a **parcela** com menor quantidade. **compra** contém um apontador para a **parcela** no topo da pilha (**raiz**) e um inteiro que representa o número de parcelas na pilha (**tamanho**).

```
typedef struct {
    /* Apontador para o topo da pilha */
    parcela *raiz;

    /* Número de parcelas na pilha */
    int tamanho;
} compra;
```

Complementarmente, foram criadas na biblioteca as funções que permitem converter uma **fatura** numa **compra**.

Implemente uma função que simula a recolha pelo cliente das parcelas na sua lista de compras. Assim, desenvolva uma função **compra_pop** que remove uma dada compra de uma pilha de compras.

Protótipo: `parcela* compra_pop(compra *c);`
Parâmetro: `c` – apontador para `compra`.
Retorno: apontador para a `parcela` removida da pilha ou `NULL` se ocorreu algum erro.