Instituto de Computação
Curso de Sistemas de Informação
Estruturas de Dados (TCC00171)

Lista de Exercícios 3

1) Considere um programa de gerenciamento comercial que manipula as informações referentes ao estoque de uma loja como um tipo abstrato de dados, no qual o tipo *Estoque* contém o ponteiro para uma lista encadeada do tipo *Produto*, que armazena as informações sobre cada tipo de produto contido no estoque. Ambos são definidos a seguir.

```
struct produto {
   int cod;                /* codigo do produto                    */
   int quant;              /* quantidade em estoque                */
   float valor;            /* valor unitario do produto            */
   struct produto *prox;   /* ponteiro para o próximo elemento     */
};
typedef struct produto Produto;

struct estoque {
   int tam;                /* número de elementos da lista         */
   Produto *lista;         /* ponteiro para lista de produtos      */
};
typedef struct estoque Estoque;
```

Para facilitar a manipulação desse tipo abstrato de dados, implemente o conjunto de funções do TAD cuja interface está descrita abaixo.

```
Estoque* cria_estoque(void);
void insere_produto(Estoque* e, int cod);
int acrescenta_quant(Estoque* e, int cod, int quant);
int decrementa_quant(Estoque* e, int cod, int quant);
int atualiza_valor(Estoque* e, int cod, float valor);
int busca_quant(Estoque* e, int cod);
float busca_valor(Estoque* e, int cod);
int remove_produto(Estoque* e, int cod);
float balanco(Estoque* e);
void libera(Estoque* e);
```

As funções são descritas a seguir.

a) A função *cria_estoque* deve alocar dinamicamente uma nova varíável do tipo *Estoque*, inicilizar o campo *tam* com o valor 0 (pois a lista de produtos ainda está vazia) e o campo *lista* com NULL, retornando um ponteiro para a nova variável, ou NULL, se não for possível alocar memória.

b) A função *insere_produto* recebe como parâmetro o ponteiro *e* para uma variável *Estoque*, o inteiro *cod* representando o código de um produto. A função deve alocar dinamicamente uma nova variável *Produto*, que deve ser inicializada com o código fornecido como parâmetro, quantidade 0 e valor 0. Além disso, o novo produto deve ser inserido na lista e o valor do campo *tam* deve ser incrementado.

c) A função *acrescenta_quant* recebe como parâmetros o ponteiro *e* para a variável *Estoque*, o inteiro *cod* representando o código de um produto e o inteiro *q*. Se o produto for encontrado, o campo *quant* deve ser acrescido do valor fornecido como parâmetro e a função deve retornar 1. Se o produto não for encontrado, a função deve retornar 0.

d) A função *decrementa_quant* recebe como parâmetros o ponteiro *e* para a variável *Estoque*, o inteiro *cod* representando o código de um produto e o inteiro *q*. Se o produto for encontrado e a quantidade em estoque (campo *quant*) for maior que *q*, a quantidade em estoque deve ser decrementada de *q* e a função deve retornar o próprio valor *q*. Se o produto for encontrado, mas a quantidade em estoque (campo *quant*) for menor que *q*, a quantidade em estoque deve ser zerada (*quant* recebe 0) e a função deve retornar o valor inicial de *quant*. Se o produto não for encontrado, a função deve retornar 0.

e) A função *atualiza_valor* recebe como parâmetros o ponteiro *e* para a variável *Estoque*, o inteiro *cod* representando o código de um produto e o float *valor*. Se o produto for encontrado, o campo *valor* deve ser atualizado recebendo o valor fornecido como parâmetro e a função deve retornar 1. Se o produto não for encontrado, a função deve retornar 0.

f) A função *busca_quant* recebe como parâmetros o ponteiro *e* para a variável *Estoque* e o inteiro *cod* representando o código de um produto. Se o produto for encontrado, a função deve retornar o valor armazenado no campo *quant* deste produto. Caso contrário, a função deve retornar -1.

g) A função *busca_valor* recebe como parâmetros o ponteiro *e* para a variável *Estoque* e o inteiro *cod* representando o código de um produto. Se o produto for encontrado, a função deve retornar o valor armazenado no campo *valor* deste produto. Caso contrário, a função deve retornar -1.

h) A função *remove_produto* recebe como parâmetros o ponteiro *e* para a variável *Estoque* e o inteiro *cod* representando o código de um produto. Se o produto for encontrado, ele deve ser removido da lista, a memória correspondente deve ser liberada, o valor do campo *tam*

deve ser decrementado e a função deve retornar 1. Se o produto não for encontrado, a função deve retornar 0.

i) A função *balanco* recebe como parâmetro o ponteiro e para a variável *Estoque* e deve retornar o valor total do estoque, ou seja, o somatório do valor em estoque de cada produto. quantidade de cada produto em estoque com o valor correspondente.

j) A função *libera* recebe como parâmetro o ponteiro e para a variável *Estoque* e efetua a liberação de todo o conjunto de variáveis alocadas dinamicamente.