1) Uma loja on-line mantém um cadastro de todos os seus clientes numa lista ordenada descrescentemente pelo total do número de compras nos 3 últimos meses. A lista é uma lista simplesmente encadeada ordenada decrescentemente por comprasUlt3Meses, onde cada nó é do tipo estruturado Elemento, descrito abaixo.

```
typedef struct cliente Cliente;
struct cliente
{       int             CPF;                    /* cpf do cliente */
        char    nome[20];           /* nome do cliente */
        int             comprasUlt3Meses; /* quantidade  de compras nos 3 últimos meses */
};
typedef struct elemento Elemento;
struct elemento
{       Cliente     *info;
        Elemento   *prox;
};
```

obs:
onde comprasUlt3Meses é número de compras feitas nos 3 últimos meses.

Assim sendo, os clientes que mais fazem compras são mantidos no início da lista.
Escreva a função *atualizaCompras*, que recebe a lista, ou seja, o endereço do primeiro nó da lista, e um CPF e incrementa (em uma unidade) o número total de compras do último trimestre desse cliente. Observe que as características da lista devem ser preservadas, o que pode implicar rearrumar a lista. Caso o cliente não exista a função retorna NULL. A função retorna o endereço do primeiro nó da lista atualizada.

3)Escreva a funcão *somaGrandesNumeros*, cujo objetivo é unicamente ler do teclado dois números inteiros muito grandes e mostrar na tela o resultado da soma desses dois números. Com esse objetivo, um número inteiro muito, muito grande pode ser armazenado numa pilha, onde cada elemento da pilha guarda um dígito desse número.
Observando que a única coisa que a função deve fazer é mostrar na tela o resultado da soma (nem os números nem a resposta precisam ser preservados), e pensando no algoritmo de soma (aquele dos seus 8 anos), escreva a função *somaGrandesNumeros* descrita no enunciado, utilizando o TAD Pilha cuja interface encontra-se abaixo. Serão necessárias 3 pilhas: 2 para os números lidos e uma para computar a resposta a ser posteriormente exibida. Novamente: nem os números nem a resposta precisam ser preservados.
obs: só iremos trabalhar com números positivos. A entrada de um número será feita teclando-se, de cada vez, um dígito do número seguido de enter. O final da entrada de cada número será indicado por um valor negativo.

Considere disponível o TAD Pilha, (de inteiros) com a seguinte interface:

```
/* pilha.h */
typedef struct pilha Pilha;
/* funcao que cria uma pilha de inteiros (digitos) e retorna o seu endereço */
Pilha *piCria (void)

/* funcao que recebe o endereço da pilha e um novo digito, inserindo-o*/
void piPush (Pilha *p, int digito);

/* funcao que recebe o end. da pilha e faz a retirada do elemento do topo da pilha, retornando-o */
int piPop (Pilha *p);

 /* funcao que recebe o endereço da pilha e retorna 1, se vazia, ou 0, em caso contrário  */
int piEhVazia (Pilha *p);

/* função que recebe o endereço de uma pilha e libera o espaço por ela utilizado */
void piLibera (Pilha *p);
```

2) Escreva a função **RECURSIVA** *copiaHomens*, que recebe o endereço do primeiro nó de uma lista simplesmente encadeada, e cria uma nova lista cujos nós são cópias dos nós de sexo masculino ('H') da lista original. A função deve retornar o endereço do primeiro nó da lista nova. Considere para representar o nó da lista encadeada o tipo estruturado No a seguir:

```
typedef struct no  No;
struct no
{       int             ident;      /* identidade da pessoa */
        char    nome[20]; /* nome da pessoa */
        char    sexo;       /* 'H' (homem) ou 'M' (mulher) */
        No      *prox;     /* ponteiro para o próximo elemento */
};
```

OBS: Caso a função apresentada não seja recursiva valerá no máximo 1.5 ponto.

4) Considere uma árvore binária de busca (ABB) com as informações dos frequentadores de uma academia, ordenada decrescentemente por idade. Escreva a função RECURSIVA *exibePorFaixaEtaria*, que recebe o endereço do nó raiz da árvore, um limite inferior (inf) e um limite superior (sup, sendo inf < sup), e exibe, em ordem crescente de idade, os dados dos frequentadores nessa faixa etária (limites excluídos). Considere o tipo estruturado NoArv abaixo, que representa um nó dessa ABB.

```
typedef struct noArv  NoArv;
struct noArv
{       char    nome[20];   /* nome da pessoa */
        int             idade;      /* idade da pessoa */
        NoArv   *esq;       /* ponteiro para subárvore da esquerda */
    NoArv       *dir;       /* ponteiro para subárvore da direita */
};
```