OPERAÇÕES EM LISTAS

1) criarLista(A) [IN: não há] [OUT: A][OBJETIVO: criar lista A vazia]

```
nA ← 0
```

2) construirLista(A,n,item) [IN: n,item] [OUT: A][OBJETIVO: construir lista com n elementos com valor item]

```
se (n>0) e (n < MaximoA) então nA ← n senão nA ← MaximoA-1;
para j de 1 até nA repita A[j] ← item
```

Consideramos que a estrutura de armazenamento de dados possa armazenar MaximoA-1 elementos.Se o valor de n (quantidade de elementos) não pertencer à faixa [1..MaximoA-1], a lista será construída com o número máximo de elementos possível.

3) esvaziarLista(A) [IN: A] [OUT: A][OBJETIVO: tornar a lista A vazia]

```
nA ← 0
```

4) obterTamanho(A,n) [IN: A] [OUT: n][OBJETIVO: obter tamanho da lista A]

```
n ← nA
```

5) verificarListaVazia(A,ok) [IN: A] [OUT: ok][OBJETIVO: obter true/false caso A seja vazia ou não]

```
se (nA = 0) então ok ← verdadeiro senão ok ← falso
```

6) obterElemento(A,p,item) [IN: A,p] [OUT: item][OBJETIVO: obter elemento que está na posição p]

```
se ((p ≥1) e (p ≤nA)) então item ← A[p] senão item ← Fantasma
```

Se o valor de p não é um valor válido, devolvemos um sinal de erro, indicado pela constante Fantasma.

/* TAD ListaInt exemplo 1*/

/* Uma lista de números inteiros é armazenada em um array de inteiros. A posição zero é reservada para guardar o comprimento da lista.*/

```
#define Maximo 6
typedef struct{
    int elemento[Maximo];
} ListaInt;

//interface
ListaInt criarListaVazia();              // construtor
ListaInt construirLista(int,int);        // construtor
ListaInt esvaziarLista(ListaInt);        // destruidor
int obterTamanho(ListaInt);              // acesso
bool verificarListaVazia(ListaInt);      // acesso
int obterElemento(ListaInt, int);        // acesso
//implementações
ListaInt criarListaVazia(){
    ListaInt listaA;    listaA.elemento[0] = 0;
    return listaA;
}
ListaInt construirLista(int n,int v){
    ListaInt listaA;
    int i,t;
    if (n==0)listaA.elemento[0] = n;
    else {
            if((n>0) && (n< Maximo))t = n; else t = Maximo-1;
            listaA.elemento[0] = t;
            for(i=1;i<=t;i++) listaA.elemento[i] = v;
    }
    return listaA;
}
```

```
ListaInt esvaziarLista(ListaInt a){
    a.elemento[0] = 0
    return a;
}
int obterTamanho(ListaInt a){        // pré-condição: a lista a está definida
    int tamanho;
    tamanho = a.elemento[0];
    return tamanho;
}
bool verificarListaVazia(ListaInt a){      // pré-condição: a lista a está definida
    bool vazia;
    if(a.elemento[0] == 0) vazia = TRUE;    else   vazia = FALSE;
    return vazia;
}
int obterElemento(ListaInt a; int p){          // pré-condição: a lista a está definida
    int valor,tamanho;
    tamanho = a.elemento[0];
    if((p>=1) && (p<=tamanho)) valor = a.elemento[p];   else valor = FANTASMA;
    return valor;
}
```

OPERAÇÕES EM LISTAS (cont)

7) buscarElemento(A,item,p) [IN: A,item] [OUT: p] [OBJETIVO: obter posição do item na lista A]

```
p ← 0;
se (nA ≠ 0) então   k ← 1;
                    enquanto ((A[k] ≠ item) e (k < nA)) faça  k ← k+1;
                    se (A[k] = item) então p ← k
```

Admitimos como pré-condição: a lista A está definida. Se o item não for encontrado na lista A, devolvemos p=0, pois a posição 0 não corresponde a nenhum elemento da lista.

8) inserirNoFim(A,novo) [IN: A, novo] [OUT: A][OBJETIVO: inserir novo elemento no final da lista A]

```
p ← nA + 1;   A[p] ←  novo;     nA ← p
```

Admitimos como pré-condição: a lista A está definida e o espaço reservado para a lista A ainda permite a inserção de um novo valor.

9) removerLocal(A,p) [IN: A, p] [OUT: A][OBJETIVO: remover o elemento da posição p]

```
se ((p ≥1) e (p ≤nA))
      então   se (p < nA) então  para j de (p+1) até nA repita  A[j-1] ← A[j];
              nA ← nA - 1
```

Admitimos como pré-condição: a lista A está definida.

Exercício 1 – fazer as implementações das 6 primeiras funções dadas no exemplo considerando a definição do tipo ListaChar a seguir.

/* Uma lista de caracteres é armazenada em um array de unsigned char identificado por vetor. Os itens da lista são armazenados partir da posição 1. O comprimento da lista é armazenado no campo tamanho. */

```
#define Maximo 6
typedef struct {
   int tamanho;
   unsigned char  vetor[Maximo];
} ListaChar;
```

Exercício 2 – fazer as implementações das 3 últimas funções (de 7 a 9) considerando a definição do tipo ListaInt dado no exemplo 1.