Considere uma lista de valores inteiros, definida pelo tipo:

```
struct no {
  int info;
  struct no* prox;
};
typedef struct no No;
```

1. Implemente uma função para inserir um novo elemento no início da lista.

```
No* insere (No* lst, int x);
```

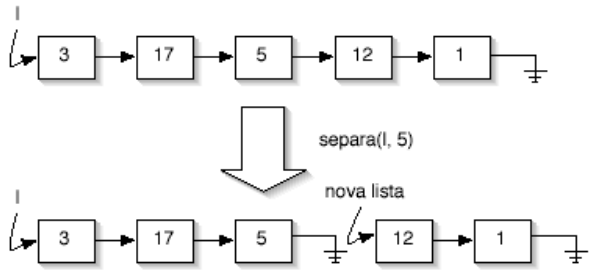
2. Implemente uma função para buscar um elemento numa lista. A função deve retornar o ponteiro do nó que contém o elemento ou NULL, se o elemento não ocorrer na lista.

```
No* busca (No* lst, int x);
```

3. Implemente uma função para retirar todas as ocorrência de um dado elemento da lista.

```
No* retira_x (No* lst, int x);
```

4. Implemente uma função que receba como parâmetro uma lista encadeada e um valor inteiro `x` e divida a lista em duas, de tal forma que a segunda lista comece no primeiro nó logo após a primeira ocorrência de `x` na lista original. A figura a seguir ilustra essa separação:



Essa função deve obedecer ao protótipo:

```
No* separa (No* lst, int n);
```

A função deve retornar um ponteiro para a segunda sub-divisão da lista original, enquanto `lst` deve continuar apontando para o primeiro elemento da primeira sub-divisão da lista.

5. Em aplicações onde há necessidade de inserir um novo elemento tanto no início quanto no final de uma lista, pode-se optar por criar uma estrutura auxiliar que representa a lista como um par de ponteiros: uma para o primeiro elemento e outro para o segundo. Uma possível estruturação, considerando o tipo `No` definido acima é:

```
struct lista {
  No* prim;    /* ponteiro para primeiro elemento */
  No* ult;     /* ponteiro para último elemento */
};
typedef struct lista Lista;
```

Desta forma, a lista passa a ser representada por um ponteiro para esta estrutura, e não apenas por um ponteiro para o primeiro nó. A cada operação sobre a lista, estes ponteiros devem ser mantidos atualizados. Considerando esta estruturação, escreva as seguintes funções:

a) Função para criar (alocar dinamicamente) uma lista vazia:

```
Lista* cria (void);
```

b) Função para inserir um elemento no início:

```
void insere_inicio (Lista* lst, int x);
```

c) Função para inserir um elemento no fim:

```
void insere_fim (Lista* lst, int x);
```

d) Função para liberar a lista (e seus elementos):

```
void libera (Lista* lst);
```