```
/*
 * Faça uma biblioteca para as definições
 * abaixo especificadas.
 */

typedef int TipoChave;
typedef int TipoValor;
struct TipoListaSimples
{
  TipoChave chave;
  TipoValor valorQualquer;
  struct TipoListaSimples *prox;
};
typedef struct TipoListaSimples TipoListaSimples;

/*============================>PROCEDIMENTOS BÁSICOS DE LISTAS COM CABEÇA
 * cabeca é ponteiro para o nó cabeça da lista
 * quando não especificado, preencha o campo valorQualquer com
 * o mesmo valor do campo chave.
 * não esqueça do assert no ponteiro do nó cabeça
 * nos casos de remoção, atente na atualização dos
 * campos adequados com NULL
 */

/*
 *-------------------------> Criacao
 * Cria nó cabeça e devolve ponteiro para ele.
 */
TipoListaSimples *criaLista();


/* -------------------------> Insercao no início
 * Insere no com valor chave igual a chave
 * no início de uma lista com cabeça.
 * Devolve endereço do nó recém inserido
 * ou NULL em caso de insucesso.
 */
TipoListaSimples *insereInicioListaSimples(TipoListaSimples *cabeca,
TipoChave chave);

/* -------------------------> Atualiza primeiro no
 * Caso o primeiro nó exista, atualiza o campo
 * valorQualquer com novoValor.
 */
void atualizaValor(TipoListaSimples *cabeca, TipoValor novoValor);

/* -------------------------> Remoção no início
 * Remove o primeiro no de uma lista com cabeca
 * caso ele exista.
 */
void removePrimeiroNo(TipoListaSimples *cabeca);

/* -----------------> Busca pelo endereço de um nó dado um valor de chave
 * Devolve ponteiro para o nó cujo valor chave é 'chave'
 * ou NULL caso este não exista.
 */
TipoListaSimples *pesquisaNo(TipoListaSimples *cabeca, TipoChave chave);
```

```
/* -------------------------> Inserção no fim de uma lista
 * Insere nó no fim de uma lista com cabeça.
 * Devolve endereço do nó ou NULL em caso de
 * insucesso.
 */
TipoListaSimples * insereFimListaSimples(TipoListaSimples *cabeca,
TipoChave chave);


/* -------------------------> Remoção último nó
 * Remove o último nó de uma lista com cabeca
 * caso ele exista.
 */
void removeUltimoNo(TipoListaSimples *cabeca);


/* -------------------------> Remove nó por valor de chave
 * Remove nó cujo valor chave seja igual a 'chave'
 * Mantêm lista inalterada caso este não exista.
 */
void removeNo(TipoListaSimples *cabeca, TipoChave chave);


/* -------------------------> Remove todos nós
 * Remove TODOS os nos da lista exceto o nó cabeça.
 */
void liberaNos(TipoListaSimples *cabeca);

/* -------------------------> Remove todos nós
 * Remove TODOS os nós da lista em que o nó cabeça encontra-se
 * armazenado no endereço *cabeca. Também remove o nó cabeça.
 */
void liberaTudo(TipoListaSimples **cabeca);


/*=========================>PROCEDIMENTOS ESPECÍFICOS DE LISTAS COM CABEÇA
/* -------------------------> Cria cópia
 * Cria uma nova lista cujos nós têm os mesmos
 * valores da lista dada. Devolve o ponteiro para
 * o cabeça da nova lista.
 */
TipoListaSimples * copiaListas(TipoListaSimples *cabeca);

/* -------------------------> Cria cópia
 * Calcula a interseção entre as duas listas
 * dadas e insere tais nós numa (nova) terceira
 * lista. Devolve o ponteiro para o cabeça da nova lista.
 */
TipoListaSimples * intersecaoListas(TipoListaSimples *cabeca1,
TipoListaSimples *cabeca2);
```

```
/* -------------------------> Insere ordenado
 * Caso a lista dada não seja vazia, ASSUMA que ela está em ordem
crescente por valor chave.
 * Ou seja, o valor chave do primeiro nó é menor ou igual ao do segundo e
assim sucessivamente.
 * Você deve inserir um novo nó com valor chave igual a chave de forma
que a lista mantenha-se
 * ordenada.
 */
void insereOrdenado(TipoListaSimples *cabeca, TipoChave chave);

/* -------------------------> Ordena
 * Devolve o nó cabeça de uma nova lista
 * com os mesmos valores da lista dada porém
 * na nova lista os valores são ordenados
 */
TipoListaSimples *ordenaLista(TipoListaSimples *cabeca);

/* -------------------------> Remove/Insere
 * Remove último nó da segunda lista (caso ele exista).
 * Insere um nó na última posição da primeira lista.
 * O novo nó deve ter os mesmos valores do nó removido.
 */
void insereRemove(TipoListaSimples *cabeca1,TipoListaSimples *cabeca2);

/* -------------------------> Transplanta Nó.
 * Desconecta o último nó da segunda lista (caso ele exista).
 * Conecta tal nó na última posição da primeira lista.
 * NOTE QUE A REGIÃO DE MEMÓRIA DO NÓ NÃO MUDA!
 */
void transplantaNo(TipoListaSimples *cabeca1, TipoListaSimples *cabeca2);
```