```c
#include <stdio.h>

#include <malloc.h>
typedef int telem; /* tipo base da lista */
typedef struct no {
        telem dado; /* campo da informação */
        struct no* prox; /*campo do ponteiro para o próximo nó*/
} tno; /* tipo do nó */

typedef tno* tlista; /* tipo lista */

/*OPERAÇÕES SOBRE LISTAS ENCADEADAS*/

/*1) Criação da lista vazia*/

void criar(tlista *L){
     *L = NULL;
}

void imprimir(tlista L) {
     tlista p;

   for (p = L; p != NULL; p = p->prox)
      printf ("\n Elemento: %d\n", p->dado);
}

/*2) Verificar se a lista está vazia*/

int vazia(tlista L){
    return (L == NULL);
}

/*3) Obter o tamanho da lista*/

int tamanho(tlista L){
    tlista p = L;
    int n = 0;
    while (p != NULL) {
          p = p->prox;
          n++;
          }
    return n;
}

/*4) Obter o valor do elemento de uma posição dada*/

int elemento(tlista L, int pos, telem *elem){
    /* O parâmetro elem irá receber o elemento encontrado */
    /* Retorna 0 se a posição for inválida. Caso contrário, retorna 1 */
   tlista p = L;
   int n = 1;
   if (L == NULL) return 0; /* erro: lista vazia */
   while ((p != NULL) && (n < pos)) {
         p = p->prox;
         n++;
         }
   if ((p == NULL)||(pos<1)) return 0; /* erro: posição inválida */
   *elem = p->dado;
   return *elem;
}
```

```
/*5) Obter a posição de elemento cujo valor é dado*/

int posicao(tlista L, telem valor){
    /* Retorna a posição do elemento ou 0 caso não seja encontrado */
    if (L != NULL) {
    tlista p = L;
    int n = 1;
    while (p != NULL) {
          if (p->dado == valor) return n;
          p = p->prox;
          n++;
          }
    }
    return 0;
}

/*6) Inserir um elemento na lista, dado a sua posição*/

int inserir(tlista *L, int pos, telem valor){
    /* Retorna 0 se a posição for inválida ou se a lista estiver cheia */
    /* Caso contrário, retorna 1 */
    tlista p, novo;
    int n;
    /* inserção em lista vazia */
    if (*L == NULL) {
       if (pos != 1) return 0; /* erro: posição inválida */
         novo = (tlista) malloc(sizeof(tno));
       if (novo == NULL) return 0; /* erro: memória insuficiente */
       novo->dado = valor;
       novo->prox = NULL;
       *L = novo;
       return 1;
    }  else {
       /* inserção na primeira posição em lista não vazia */
       if (pos == 1) {
         novo = (tlista) malloc(sizeof(tno));
         if (novo == NULL) return 0; /* erro: mem¢ria insuficiente */
         novo->dado = valor;
         novo->prox = *L;
         *L = novo;
         return 1;
         } else {
             /* inserção após a primeira posição em lista não vazia */
             p = *L;
             n = 1;
             while ((n < pos-1) && (p != NULL)) {
                   p = p->prox;
                   n++;
             }
             if (p == NULL) return 0; /* erro: posição inválida */
             novo = (tlista) malloc(sizeof(tno));
             if (novo == NULL) return 0; /* erro: memória insuficiente */
             novo->dado = valor;
             novo->prox = p->prox;
             p->prox = novo;
             return 1;
         }
         }
}

/*7) Remover um elemento de uma determinada posição*/
```

```
int remover(tlista *L, int pos, telem *elem){
    /* O parâmetro elem irá receber o elemento encontrado */
    /* Retorna 0 se a posição for inválida. Caso contrário, retorna 1 */
    tlista a, p;
    int n;
    if (vazia(*L)) return 0; /* erro: lista vazia */
    p = *L;
    n = 1;
    while ((n<=pos-1) && (p!=NULL)) {
          a = p;
          p = p->prox;
          n++;
    }
    if (p == NULL) return 0; /* erro: posição inválida */
    *elem = p->dado;
    if (pos == 1)
    *L = p->prox;
    else
        a->prox = p->prox;
    free(p);
    return(1);
}


int main(int argc, char *argv[])
{
    tlista L1;
    int i, valor;
    telem elem;

    criar(&L1);

    inserir(&L1, 1, 7);
    imprimir(L1);
    printf("\n Tamanho da Lista: %d\n", tamanho(L1));

    inserir(&L1, 2, 34);
    imprimir(L1);
    printf("\n Tamanho da Lista: %d\n", tamanho(L1));

    inserir(&L1, 3, 15);
    imprimir(L1);
    printf("\n Tamanho da Lista: %d\n", tamanho(L1));

    inserir(&L1, 2, 16);
    imprimir(L1);
    printf("\n Tamanho da Lista: %d\n", tamanho(L1));

    remover(&L1, 2, &elem);
    imprimir(L1);
    printf("\n Tamanho da Lista: %d\n", tamanho(L1));

    if (elemento(L1, 2, &elem))
        printf("\n Elemento: %d\n", elem);
        else {
              printf("\n Localizacao invalida");
        }

    if (posicao(L1, 34))
        printf("\n Posicao: %d\n", posicao(L1, 34));
```

```
        else {
              printf("\n Localizacao invalida");
              }

    getchar();
    return 0;
}
```