```
STRUCT ELEMENTO{
  INT DADO;
STRUCT ELEMENTO *PROX; };
TYPEDEF STRUCT ELEMENTO
ELEMENTO;
TYPEDEF ELEMENTO* LISTA;
LISTA* CRIAR_LISTA(){
  LISTA *LI = (LISTA*)
MALLOC(SIZEOF(LISTA));
   IF(LI != NULL) *LI = NULL;
ELSE{ RETURN NULL;
RETURN LI; }
VOID LIBERAR_LISTA(LISTA *LI){
IF(LI != NULL){
ELEMENTO *NO;
   WHILE( (*LI) != NULL ){
NO = *LI; *LI = (*LI)->PROX;
FREE(NO); }
    FREE(LI);
INT TAMANHO_LISTA(LISTA *LI){

IF(LI == NULL) RETURN 0;
 INT CONT = 0
 ELEMENTO *NO = *LI;
 WHILE(NO != NULL){
   CONT++;
NO = NO->PROX; }
 RETURN CONT;}
INT LISTA_VAZIA(LISTA *LI){
IF(LI == NULL) RETURN 1;
IF(*LI == NULL) RETURN 1;
RETURN 0;}
INT LISTA_CHEIA(LISTA *LI)
RETURN 0;
 ELEMENTO*
CRIAR_ELEMENTO(){
 ELEMENTO *NO = (ELEMENTO*)
 MALLOC(SIZEOF(ELEMENTO));
 IF(NO == NULL) RETURN 0;
 RETURN NO;}
INT INS_LIST_INI(LISTA *LI, INT
DADO){

IF(LI == NULL) RETURN 0;
 ELEMENTO *NO:
 NO = CRIAR_ELEMENTO();
 NO->DADO = DADO;
 NO->PROX = NULL;
 *LI = NO;
 RETURN 1;}
```

```
INT INSER_LIST_Final(LISTA *LI,
INT DADO){
IF(LI == NULL) RETURN 0;
 ELEMENTO *NO;
NO = CRIAR_ELEMENTO();
NO->DADO = DADO;
 NO->PROX = NULL;
 IF( (*LI) == NULL ) *LI = NO;
   ELSÉ{
     ELEMENTO *AUX;
     AUX = *LI;
WHILE(AUX->PROX != NULL)
     AUX = AUX->PROX;
AUX->PROX = NO;
 RÉTURN 1:}
INT INSER_LIST_ORD(LISTA *LI,
INT DADO){
IF(LI == NULL) RETURN 0;
 ELEMENTO *NO;
NO = CRIAR_ELEMENTO();
NO->DADO = DADO;
 IF( (*LI) == NULL ){
    NO->PROX = NULL;
    *LI = NO:
    ELSE{
      ELEMENTO *ANTERIOR,
                                         DADOX
*ATUAL;
      ATUAL = *LI;
       WHILE(ATUAL != NULL &&
ATUAL->DADO < DADO){
    ANTERIOR = ATUAL;
         ATUAL = ATUAL->PROX;
    IF( ATUAL == (*LI) ){
       NO->PROX = (*LI);
                                         POS)
       *LI = NO;
        ELSE{
          NO->PROX = ATUAL;
          ANTERIOR->PROX = NO;
                                         DADO)
RÉTURN 1;}
```

```
INT REMOV_LISTA_INICIO(LISTA *LI){
 IF(LI == NULL) RETURN 0;
IF((*LI) == NULL) RETURN 0;
 ELEMENTO *NO;
 NO = *LI;
*LI = NO->PROX;
 FREE(NO);
 RETURN 1:}
INT REMOV_LISTA_FINAL(LISTA *LI){
 IF(LI == NULL) RETURN 0;
IF((*LI) == NULL) RETURN 0;
ELEMENTO *ANTERIOR, *NO;
 NO = *LI;
 WHILE(NO->PROX != NULL){
    ANTERIOR = NO;
    NO = NO->PROX;
 IF(NO == (*LI)) *LI = NO->PROX;
     ELSE
       ANTERIOR->PROX = NO->PROX;
 FREE(NO);
 RETURN 1;}
INT REM_LIST_MEIO(LISTA *LI, INT
DADO){

IF(LI == NULL) RETURN 0;
 IF((*LI) == NULL) RETURN 0;
ELEMENTO *ANTERIOR, *NO;
 WHILE(NO != NULL && NO->DADO !=
    ANTERIOR = NO;
    NO = NO->PROX;
IF(NO == NULL) RETURN 0;
IF(NO == (*LI)) *LI = NO->PROX;
ELSE ANTERIOR->PROX =
NO->PROX;
FREE(NO);
RETURN 1:3
INT BUS_LISTA_POS(LISTA *LI, INT
 /*** IMPLEMENTAR ***/
INT BUSC_LIST_DADO(LISTA *LI, INT
 /*** IMPLEMENTAR ***/
```

```
INT REM_LIST_MEIO(LISTA *LI,
STRUCT ELEMENTO(
                                      INT INSER_LIST_ORD(LISTA *LI,
                                                                            INT DADO){
 INT DADO:
                                      INT DADOX
 STRUCT ELEMENTO *PROX; };
                                       IF(LI == NULL) RETURN 0;
                                                                             IF(LI == NULL) RETURN 0;
                                                                            IF((*LI) == NULL) RETURN 0;
ELEMENTO *NO;
                                       ELEMENTO *NO:
TYPEDEF STRUCT ELEMENTO
                                       NO = CRIAR_ELEMENTO();
NO->DADO = DADO;
ELEMENTO;
TYPEDEF ELEMENTO* LISTA;
                                                                             NO = *LI;
                                                                             IF(NO->DADO == DADO){
                                       IF( (*LI) == NULL ){
   NO->PROX = NO;
LISTA* CRIAR_LISTA(){
LISTA*LI = (LISTA*)
                                                                              IF (NO->PROX == NO){
MALLOC(SIZEOF(LISTA));
                                         *LI = NO;
                                                                                *LI = NULL;
IF(LI != NULL) *LI = NULL;
ELSE RETURN NULL;
                                                                                FREE(NO):
                                                                                RETURN 1;
                                         ELSE{
 RETURN LI;}
                                           IF((*LI)->DADO > DADO){
INT LIBERAR_LISTA(LISTA *LI){
IF(LI == NULL) RETURN 0;
                                             ELEMENTO *AUX:
                                                                               FLSE(
                                                                                 ELEMENTO *ULTIMO;
                                             AUX = *LI:
 IF((*LI) != NULL){
                                             WHILE(AUX->PROX != (*LI))
                                                                                 ULTIMO = *LI;
   ELEMENTO *NO. *AUX:
                                              AUX = AUX->PROX:
                                                                                 WHILE(ULTIMO->PROX !=
                                             AUX->PROX = NO;
                                                                           (*LI))
   WHILE( NO->PROX != (*LI) ){
                                             NO->PROX = *LI;
                                                                                   ULTIMO =
    AUX = NO;
NO = NO-> PROX;
                                             *I I = NO:
                                                                            UI TIMO->PROX:
                                                                                 ULTIMO->PROX =
                                                                            NO->PROX;
     FREE(AUX);
                                           ÉLEMENTO *ANTERIOR,
                                                                                 *I I = NO->PROX:
                                      *ATUAL:
   FREE(NO);
                                          ANTERIOR = *LI;
                                                                                 FREE(NO);
                                           ATUAL = ANTERIOR->PROX;
                                                                                 RETURN 1;
 FREE(LI):
                                      WHILE(ATUAL != (*LI) &&
ATUAL->DADO < DADO){
                                                                                }
 RETURN 1;}
INT TAMANHO_LISTA(LISTA *LI){
                                             ANTERIOR = ATUAL;
                                                                             ELEMENTO *ANTERIOR;
                                                                            ANTERIOR = NO;
NO = NO->PROX;
IF(LI == NULL || (*LI) == NULL)
                                             ATUAL = ATUAL->PROX;
RETURN 0:
 INT CONT = 0;
                                           ANTERIOR->PROX = NO;
                                                                             WHILE(NO != (*LI) &&
 ELEMENTO *NO;
                                           NO->PROX = ATUAL;
                                                                            NO->DADO != DADO){
 NO = *LI:
                                                                              ANTERIOR = NO:
                                       RETURN 1;}
                                                                              NO = NO->PROX;
 DO{
   CONT++;
NO = NO->PROX;
                                      INT
                                                                            IF(NO == (*LI)) RETURN 0;
ANTERIOR->PROX =
                                      REMOVER LISTA INICIO(LISTA
 }WHILE(NO != (*LI));
                                       *LI){
RETURN CONT;}
INT LISTA_VAZIA(LISTA *LI){
                                       IF(LI == NULL) RETURN 0;
                                                                            NO->PROX;
                                       IF((*LI) == NULL) RETURN 0;
                                                                            FREE(NO):
 IF(LI == NULL) RETURN -1;
IF(*LI == NULL) RETURN 1;
                                       IF ((*LI)->PROX == (*LI)){
                                                                             RETURN 1;}
                                         FREÉ(*LI);
                                                                            BUSCAR_LISTA_POSICAO(LIST
RÈTURN 0;}
ELEMENTO* CRIAR_ELEMENTO(){
                                         *LI = NULL:
                                         RETURN 1;
                                                                            A *LI, INT POS){
                                                                            IF(LI == NULL || (*LI) == NULL ||
 ELEMENTO *NO = (ELEMENTO*)
                                       ELEMENTO *NO. *AUX:
MALLOC(SIZEOF(ELEMENTO));
                                                                            POS <= 0)
                                                                               RETURN -1;
 IF(NO == NULL) RETURN 0;
RETURN NO;}
INT INSERIR_LISTA_INICIO(LISTA
                                                                            INT I = 1;
ELEMENTO *NO;
                                       AUX = *LI;
                                       WHILE(AUX->PROX != (*LI)){
                                         AUX = AUX->PROX;
LI, INT DADO){
                                                                             NO = *LI;
               RETURN 0;
                                                                             WHILE( NO->PROX != (*LI) && I
 IF(LI == NULL)
 ELEMENTO *NO:
                                       AUX->PROX = NO->PROX:
                                                                            < POS){
NO = NO->PROX;
 NO = CRIAR_ELEMENTO();
                                       *LI = NO->PROX;
 NO->DADO = DADO;
                                       FREE(NO);
IF((*LI) == NULL){
NO->PROX = NO;
                                       RETURN 1;}
                                                                             IF(I != POS) RETURN -1;
   *LI = NO;
                                      REMOVER_LISTA_FINAL(LISTA
                                                                            RETURN NO->DADO;}
  ELSE(
                                      *LI){
   ELEMENTO *AUX;
                                                                            BUSCAR_LISTA_DADO(LISTA
                                       IF(LI == NULL) RETURN 0;
                                       IF((*LI) == NULL) RETURN 0;
IF ((*LI)->PROX == (*LI)){
    AUX = *LI;
                                                                            *LI, INT DADO){
                                                                            IF(LI == NULL || (*LI) == NULL)
   WHILE(AUX->PROX != (*LI))
     AUX = AUX->PROX;
                                                                               RETURN -1;
                                         FREÉ(*LI);
                                                                            INT I = 1;
ELEMENTO *NO;
    AUX->PROX = NO;
                                         *LI = NULL:
    NO->PROX = *LI:
                                         RETURN 1;
    *LI = NO;
                                                                             NO = *LI;
                                                                             WHILE( NO->PROX != (*LI) &&
                                       ELEMENTO *ANTERIOR, *NO;
 RETURN 1:3
                                                                            NO->DADO != DADO){
                                       NO = *LI:
INT INSERIR_LISTA_FINAL(LISTA
                                       WHILE(NO->PROX != (*LI)){
                                                                               NO = NO->PROX;
*LI, INT DADO){
                                         ANTÈRIOR = NO;
IF(LI == NULL) RETURN 0;
ELEMENTO *NO;
                                         NO = NO->PROX;
                                                                            IF(NO->DADO != DADO)
 NO = CRIAR_ELEMENTO();
                                       ANTERIOR->PROX = NO->PROX;
                                                                            RETURN -1;
 NO->DADO = DADO;
                                       FREE(NO);
                                                                            RETURN I:3
 IF( (*LI) == NULL ){
                                       RETURN 1;}
                                                                            INT IMPRIMIR_LISTA(LISTA *LI){
   NO->PROX = NO;
                                                                             IF(LI == NULL) RETURN 0;
                                                                            IF((*LI) == NULL) PRINTF("
   *LI = NO;
                                                                            LISTA VAZIA!");
  ELSE{
                                                                             ELEMENTO *NO;
   ELEMENTO *AUX;
                                                                             NO = (*LI);
                                                                             WHILE(NO->PROX != (*LI)){
   AUX = *LI;
                                                                              PRINTF(" %D", NO->DADO);
   WHILE(AUX->PROX != (*LI))
                                                                              NO = NO->PROX:
     AUX = AUX->PROX;
    AUX->PROX = NO;
                                                                             PRINTF(" %D", NO->DADO);
   NO->PROX = *LI;
                                                                             RETURN 1;}
 RETURN 1;}
```