/\* FUNCOES LISTA SEM CABECALHO \*/

typedef struct lnode \*List;

typedef struct lnode {

int info;

List next;

}List\_node;

List cria\_lista(void){

return NULL

}

List destroi\_lista (List lista){

List temp\_ptr;

while (lista\_vazia (lista) == 0) {

temp\_ptr = lista;

lista = lista->next;

free (temp\_ptr);

}

return NULL;

}

int lista\_vazia(List lista){

if (lista == NULL)

return 1;

else

return 0;

}

void procura\_lista (List lista, ITEM\_TYPE chave, List \*ant, List \*actual){

\*ant = NULL;

\*actual = lista;

while ((\*actual) != NULL && (\*actual)->info < chave){

\*ant = \*actual;

\*actual = (\*actual)->next;

}

if ((\*actual) != NULL && (\*actual)->info != chave)

\*actual = NULL; /\* Se elemento não encontrado\*/

}

void elimina\_lista (List lista, ITEM\_TYPE it){

List ant;

List actual;

ant = NULL

actual = lista;

while((actual != NULL)&&(actual->info != it)){

ant = actual;

actual = actual -> next;

}

if (actual != NULL){

ant -> next = act -> next;

if (actual==lista){

lista = lista->next;

}

free(actual);

}

return lista;

}

void imprime\_lista (List lista){

List l = lista;

while (l){

printf("%d ", l->info);

l=l->next;

}

}

List insereLista(List lista, int it){ /\*insere no fim da lista \*/

List actual;

List ant;

List novo;

novo = (List) malloc(sizeof(List\_node));

novo->info = it;

novo->next = NULL;

ant = NULL;

actual = lista;

while(actual != NULL){

ant = actual;

actual = actual->next;

}

if (ant == NULL)

lista = novo;

else

ant->next = novo;

return lista;

}

List insereLista(List lista, int it){ /\*insere no inicio \*/

List novo = (List) malloc (sizeof(List\_node));

if(novo != NULL){

novo -> info = it;

novo -> next = lista;

lista = novo;

}

return lista;

}

int main(){

List lista;

lista = cria\_lista();

lista = destroi\_lista(lista);

lista = elimina\_lista(lista);

return 0;

}