I

void removeElemento(DLista \*lst, char \*s)

{

Ligacao \*x;

for (x = lst->primeiro; x != NULL; x = x->proximo)

{

if (strcmp(x->text, s) == 0)

{

break; /\*parar quando x->text == s, ou seja temos de remover x\*/

}

}

/\* se elemento a remover for a head \*/

if (x->anterior==NULL){

lst->primeiro = x->proximo;

lst->primeiro->anterior = NULL;

}

else if (x->proximo==NULL){

/\* se elemento a remover for o ultimo \*/

lst->ultimo = x->anterior;

lst->ultimo->proximo=NULL;

} else{

/\*se elemento a remover for no meio\*/

x->anterior->proximo = x->proximo;

x->proximo->anterior = x->anterior;

}

free(x->text);

free(x);

}

Análise Pior caso:

O(n) -> ter de percorrer a doubly linked list toda, n = tamanho da doubly linked list

------------------------------------------------------------------------------

II

Indice | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |

Elementos| 13,39,0,26 | X | 28,15 | X | X | X | X | 20 | X | X | 36 | X | 25,12 |

Faria com que a head de cada lista de cada index da tabela contivesse um ponteiro para

o node final da lista, nesse caso seria facil colocar o elemento logo após o ultimo

elemento da lista, usando uma funcao insertEnd em tempo constante O(1).