Alocação Encadeada- Nós de cabeçalho- Exercício

Implemente, no TAD LISTA_ENC_NC_ORD, a seguinte operação:

LISTA_ENC_NC_ORD concatenar (LISTA_ENC_NC_ORD, LISTA_ENC_NC_ORD);

a qual recebe duas listas e retorna uma lista resultante da concatenação das mesmas. ***OBS. A lista resultante não deve apresentar elementos com mesmo campo inf.***

Listas Circulares

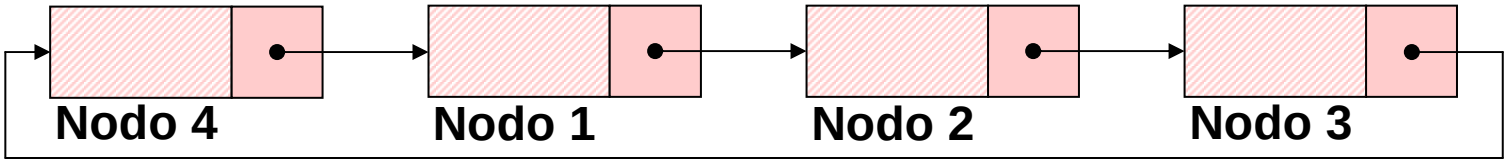
Listas lineares encadeadas possuem alguma grande deficiência. Qual?

Dado um ponteiro ***p*** para um nodo de uma lista linear encadeada, não podemos atingir nenhum nodo que antecede o apontado por ***p***.

Contudo, se fizermos uma pequena alteração na estrutura de lista que temos trabalhado, fazendo com que o campo ***next*** do último nodo ao invés de conter ***NULL*** armazene o endereço do primeiro nodo da lista.

Listas Circulares

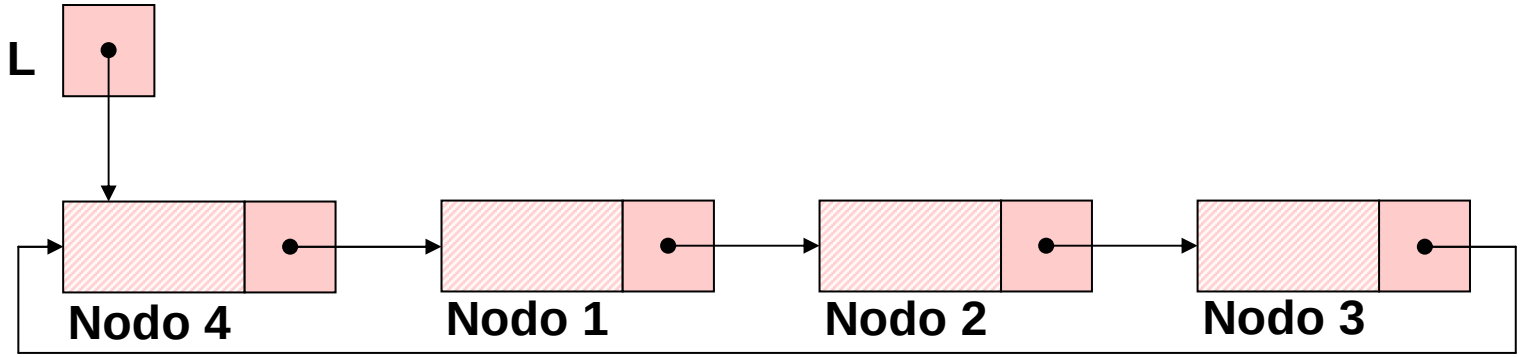
Este tipo de lista é denominado lista circular, possuindo a forma abaixo:



Uma pergunta surge: para qual elemento apontar para que se tenha uma referência a lista circular?

Listas Circulares

Uma convenção útil é fazer com que o ponteiro esterno para a lista circular aponte para o último elemento.



Pois desta forma se tem acesso direto ao último e primeiro elemento.

Listas Circulares

Como podemos observar a definição do TAD LISTA_CIRCULAR é praticamente a mesma do TAD LISTA_ENC, apenas algumas pequenas alterações são necessárias nas operações.

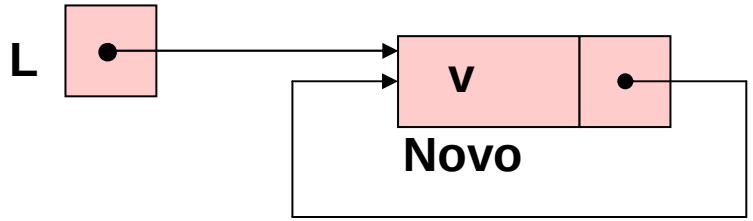
```
typedef struct nodo
{
   int inf;
   struct nodo * next;
}NODO;
typedef NODO * LISTA_CIRCULAR;
void cria_lista (LISTA_CIRCULAR *);
int eh_vazia (LISTA_CIRCULAR);
int tam (LISTA_CIRCULAR);
void ins (LISTA_CIRCULAR *, int, int);
int recup (LISTA_CIRCULAR, int);
void ret (LISTA_CIRCULAR *, int);
```

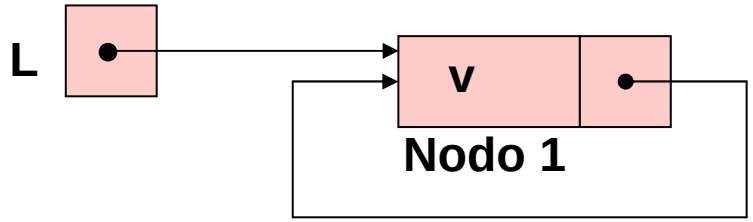
```
void cria_lista (LISTA_CIRCULAR *pl)
{
   *pl=NULL;
}

int eh_vazia (LISTA_CIRCULAR l)
{
   return (l == NULL);
}
```

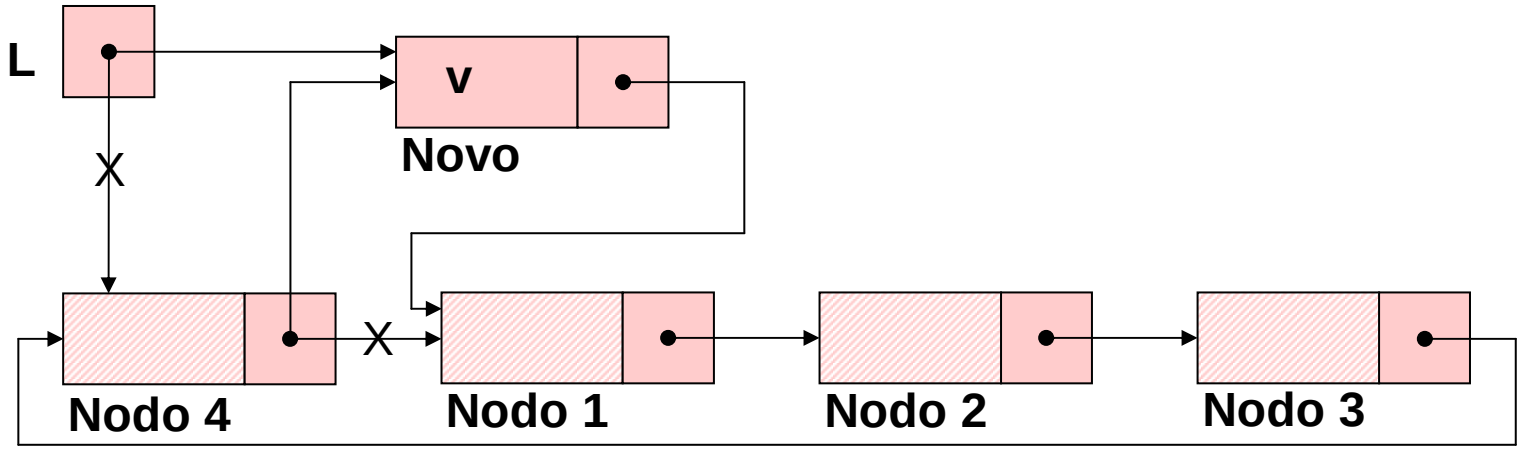
```
int tam (LISTA_CIRCULAR l)
{
   if (l==NULL)
      return (0);
   else
   {
      LISTA_CIRCULAR aux;
      int cont;
      for (cont=1, aux=l->next; aux!=l ; cont++)
         aux = aux->next;
      return (cont);
   }
}
```

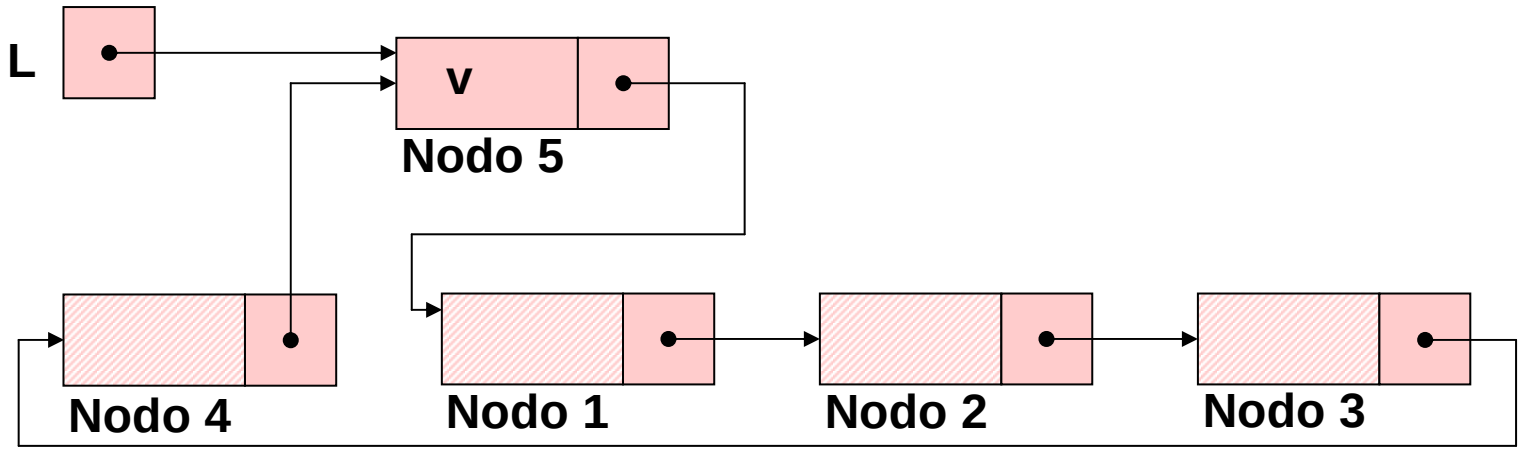
Esquema do processo da inserção de um nó da lista circular. (situação um)





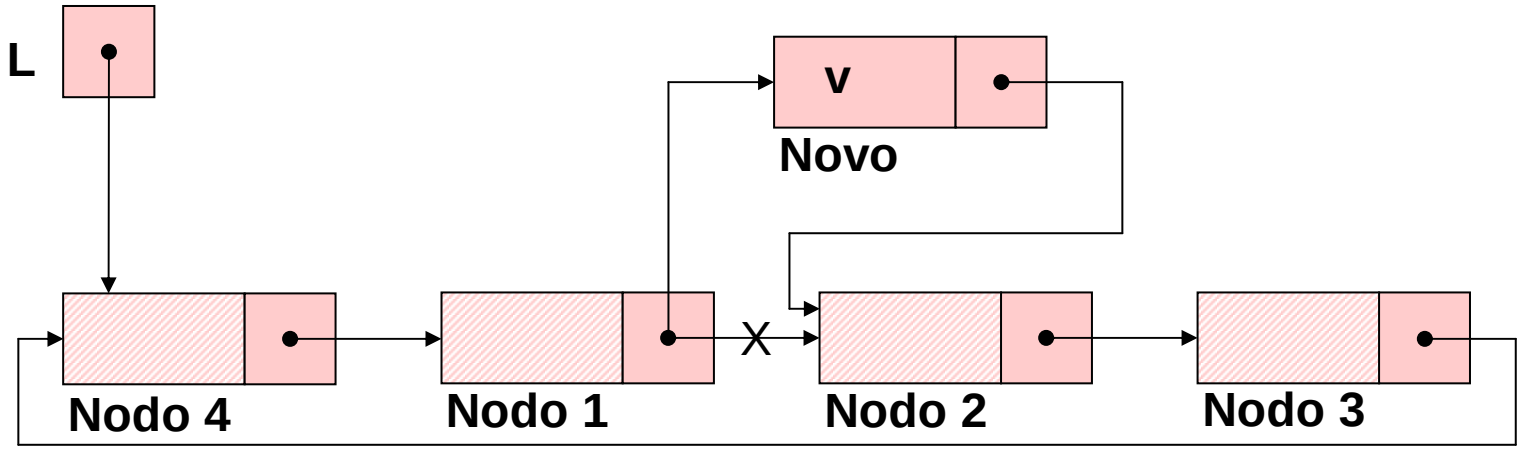
Esquema do processo da inserção de um nó da lista circular. (situação dois)

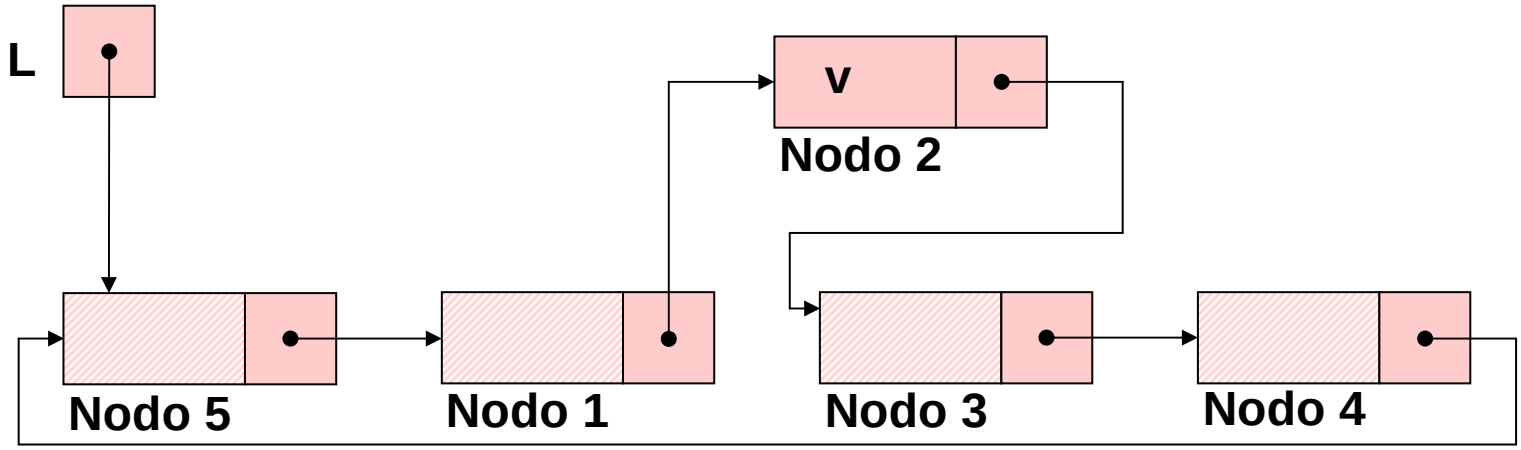




Listas Circulares

Esquema do processo da inserção de um nó da lista circular. (situação três)





```
void ins (LISTA_CIRCULAR *pl, int v, int k)
{
  NODO *novo;
  if (k < 1 || k > tam(*pl)+1)
  {
    printf ("\nERRO! Posição invalida para insercao.\n");
    exit (1);
  }
  novo = (NODO *) malloc (sizeof(NODO));
  if (!novo)
  {
    printf ("\nERRO! Memoria insuficiente!\n");
    exit (2);
  }
```

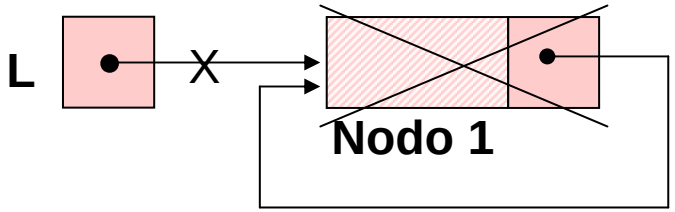
```
novo->inf = v;
if (*pl==NULL)    {
   novo->next=novo;
   *pl = novo;    }
else
{
   LISTA_CIRCULAR aux=*pl;
   if (k==tam(*pl)+1)
      *pl=novo;
   for (; k>1; aux=aux->next, k--);
   novo->next = aux->next;
   aux->next = novo;      } }
```

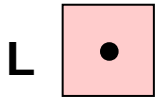
Exercício:

Podemos fazer uma operação de inserção recursiva para o TAD LISTA_CIRCULAR, implemente esta operação.

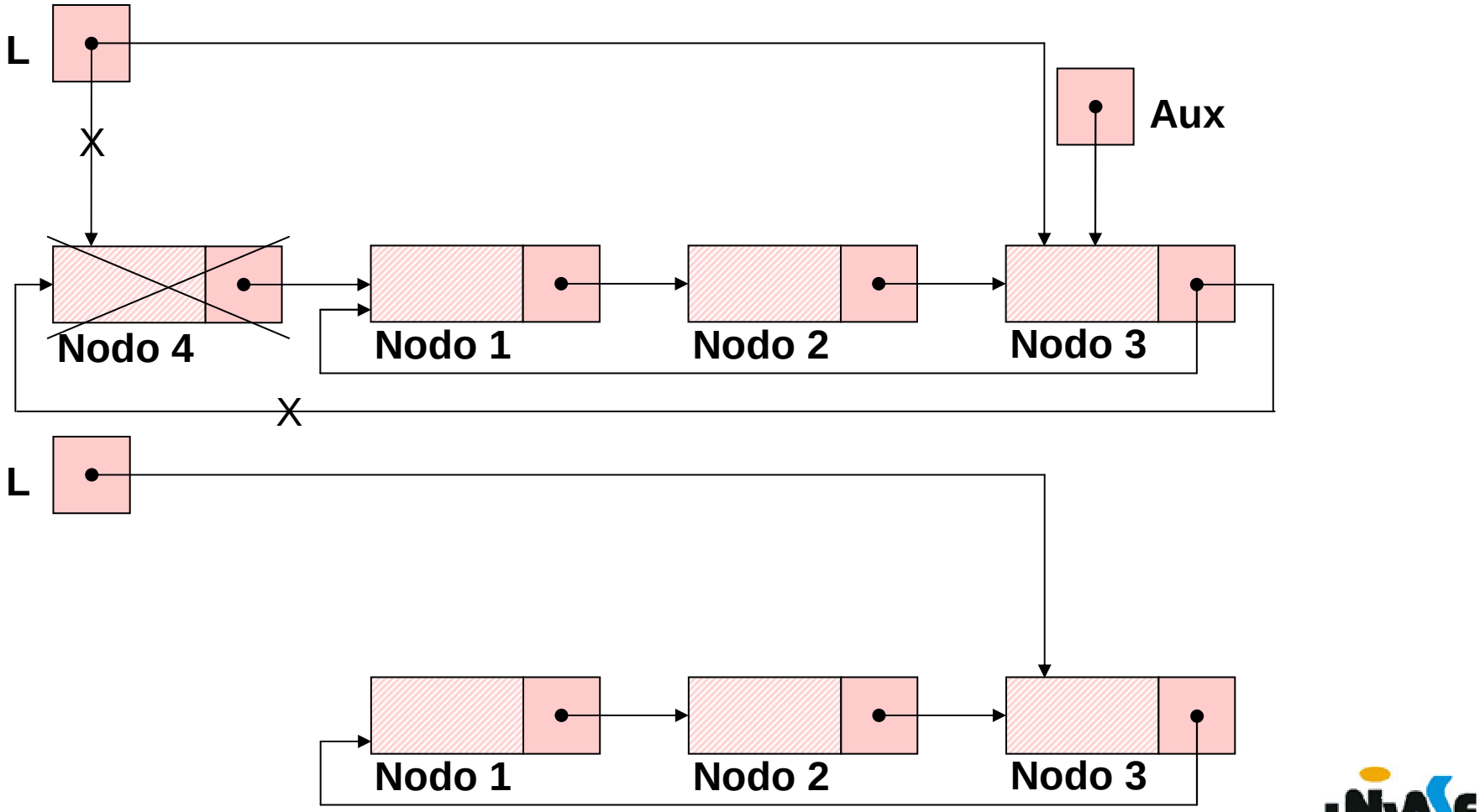
```
int recup (LISTA_CIRCULAR l, int k)
{
   if (k < 1 || k > tam(l))
   {
      printf ("\nERRO! Consulta invalida.\n");
      exit (3);
   }
   for (;k>0;k--)
      l=l->next;
   return (l->inf);
}
```

Esquema do processo da retirada de um nó da lista circular. (situação um)



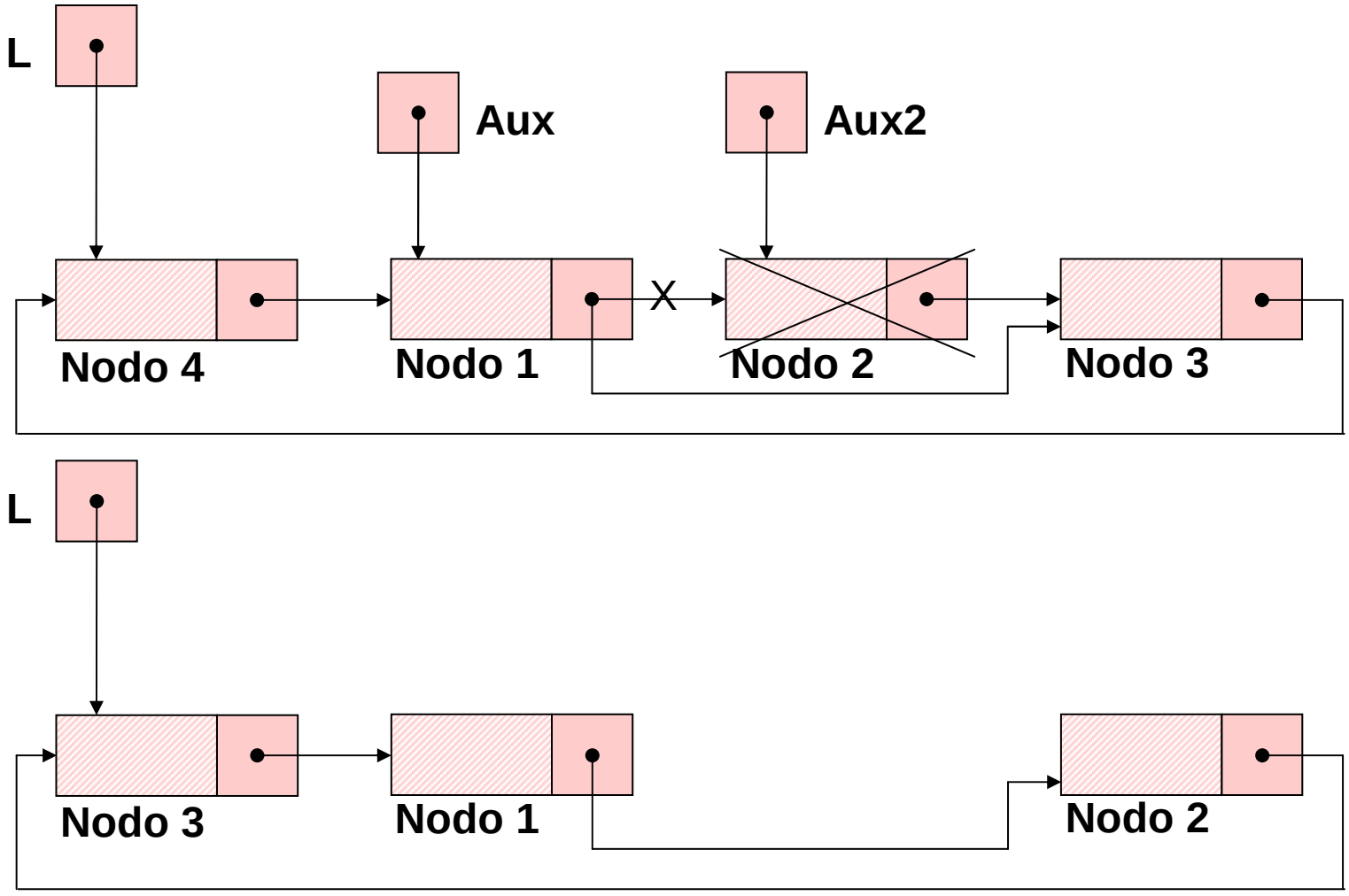


Esquema do processo da retirada de um nó da lista circular. (situação dois)



Listas Circulares

Esquema do processo da retirada de um nó da lista circular. (situação três)



```
void ret (LISTA_CIRCULAR *pl, int k)
{
   if (k < 1 || k > tam(*pl))
   {
     printf ("\nERRO! Posição invalida para retirada.\n");
     exit (4);
   }
   if (tam(*pl)==1)
   {
      free (*pl);
      *pl = NULL;
   }
```

```
    else
    {
       NODO *aux, *aux2;
       int i;
       for (aux=*pl, i=k; i>1; i--, aux=aux->next);
       aux2 = aux->next;
       aux->next = aux2->next;
       if (k==tam(*pl))
          *pl=aux;
       free (aux2);
    }
}
```

Exercício:

Podemos fazer uma operação de retirada recursiva para o TAD LISTA_CIRCULAR, implemente esta operação.

Listas Circulares – Nó de Cabeçalho

O conceito de ***nó de cabeçalho*** também pode ser empregado nas listas circulares. Por exemplo, podemos utilizar o nó de cabeçalho como um ponto de verificação para testar se a lista inteira foi atravessada.

A implementação de um TAD LISTA_CIRCULAR_COM_NC é sugerida como uma exercício de fixação.