Alocação Encadeada – Nó de cabeçalho

Com base no que foi visto implemente a operação destruir() que compõem o TAD LISTA_ENC_NC.

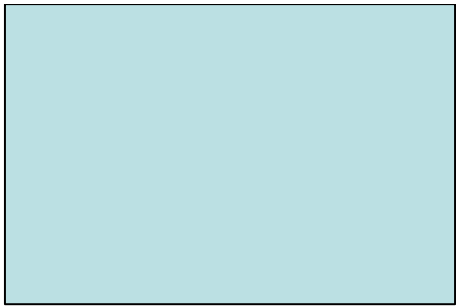
```
1  typedef struct nodo
2  {
3     int inf;
4     struct nodo * next;
5  }NODO;
6  typedef NODO * LISTA_ENC_NC;
7  void cria_lista (LISTA_ENC_NC *);
8  int eh_vazia (LISTA_ENC_NC);
9  int tam (LISTA_ENC_NC);
10 void ins (LISTA_ENC_NC, int, int);
11 int recup (LISTA_ENC_NC, int);
12 void ret (LISTA_ENC_NC, int);
13 void destruir (LISTA_ENC_NC);
```

```c
void destruir (LISTA_ENC_NC l)
{
    LISTA_ENC_NC aux;
    while (l)
    {
        aux = l;
        l = l->next;
        free(aux);
    }
}
```

```c
void destruir (LISTA_ENC_NC l)
{
    while (l)
    {
        free(l);
        l = l->next;
    }
}
```

Listas – Alocação Encadeada – Ordenada e com Nó Cabeçalho – Exercícios

Alocação Encadeada – Nó de cabeçalho



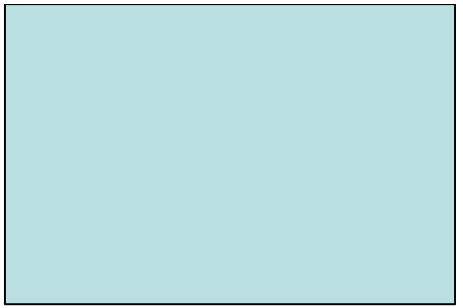
Como vimos, a única operação que requer alteração para transformarmos o TAD LISTA_ENC no TAD LISTA_ ENC_ORD é a operação de inserção, o mesmo ocorre com os TAD ´s LISTA_ENC_NC e LISTA_ENC_NC_ ORD.

Com base na operação de inserção implementada no TAD LISTA_ENC_ORD implemente a operação de inserção de um elemento no TAD LISTA_ENC_NC_ORD.

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

```c
void ins (LISTA_ENC_NC_ORD l, int v)
{
    NODO *novo;
    novo = (NODO *) malloc (sizeof(NODO));
    if (!novo)
    {
       printf ("\nERRO! Memoria insuficiente!\n");
       exit (2);
    }
    l->inf++;
    novo->inf = v;
    for (; l->next!=NULL && v>(l->next)->inf;
    l=l->next);
    novo->next = l->next;
    l->next = novo;
}
```

Alocação Encadeada – Nó de cabeçalho



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.

```
1  LISTA_ENC_NC_ORD concatenar (LISTA_ENC_NC_ORD l1,
2  LISTA_ENC_NC_ORD l2) {
3     LISTA_ENC_NC_ORD l;
4     int aux;
5     l1=l1->next;
6     l2=l2->next;
7     cria_lista (&l);
8     while (l1 || l2)
9     {
10        if (l1 && !l2)
11          aux=l1->inf;
12        else
13          if (!l1 && l2)
14             aux=l2->inf;
```

```
15              else
16                  if (l1->inf < l2->inf)
17                      aux=l1->inf;
18                  else
19                      aux=l2->inf;
20          ins (l, aux);
21          while(l1 && l1->inf==aux)
22            l1=l1->next;
23          while(l2 && l2->inf==aux)
24            l2=l2->next;
25      }
26      return (l);
27  }
```

Alocação Encadeada – Nó de cabeçalho

Com base no que foi visto implemente, utilizando recursividade, as operações ins(), recup(), ret() e destruir() que compõem o TAD LISTA_ENC_NC.

```
1   typedef struct nodo
2   {
3      int inf;
4      struct nodo * next;
5   }NODO;
6   typedef NODO * LISTA_ENC_NC;
7   void cria_lista (LISTA_ENC_NC *);
8   int eh_vazia (LISTA_ENC_NC);
9   int tam (LISTA_ENC_NC);
10  void ins (LISTA_ENC_NC, int, int);
11  int recup (LISTA_ENC_NC, int);
12  void ret (LISTA_ENC_NC, int);
13  void destruir (LISTA_ENC_NC);
```