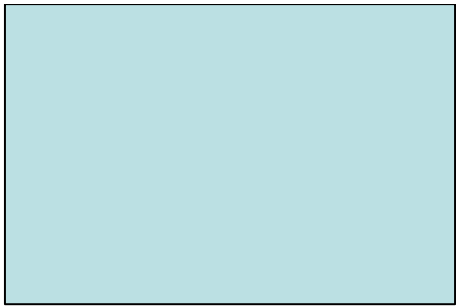
Alocação Encadeada

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

```
typedef struct nodo
{
   int inf;
   struct nodo * next;
}NODO;
typedef NODO * LISTA_ENC;
```

```
void cria_lista (LISTA_ENC *);
int eh_vazia (LISTA_ENC);
int tam (LISTA_ENC);
void ins (LISTA_ENC *, int, int);
int recup (LISTA_ENC, int);
void ret (LISTA_ENC *, int);
void destruir (LISTA_ENC);
```

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



Listas – Alocação Encadeada – Exercícios

Alocação Encadeada

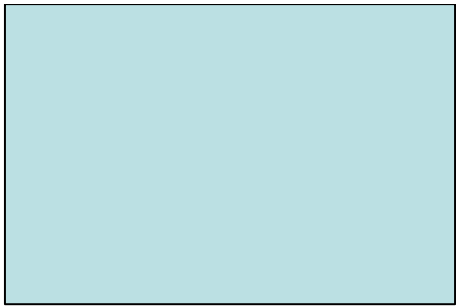
Com base no que foi visto implemente a operação tam() que compõem o TAD LISTA_ENC utilizando recursividade.

```
typedef struct nodo
{
   int inf;
   struct nodo * next;
}NODO;
typedef NODO * LISTA_ENC;
```

```
void cria_lista (LISTA_ENC *);
int eh_vazia (LISTA_ENC);
int tam (LISTA_ENC);
void ins (LISTA_ENC *, int, int);
int recup (LISTA_ENC, int);
void ret (LISTA_ENC *, int);
void destruir (LISTA_ENC);
```

UNIVASF

```
int tam (LISTA_ENC l)
{
    if (!l)
        return (0);
    else
        return (1 + tam(l->next));
}
```



Alocação Encadeada

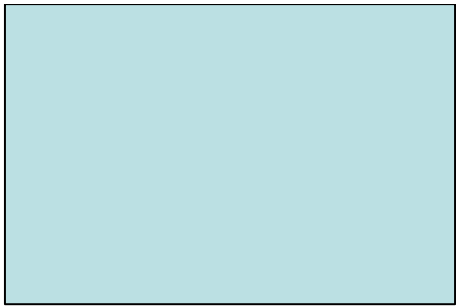
Com base no que foi visto implemente a operação recup() que compõem o TAD LISTA_ENC utilizando recursividade.

```
typedef struct nodo
{
   int inf;
   struct nodo * next;
}NODO;
typedef NODO * LISTA_ENC;
```

```
void cria_lista (LISTA_ENC *);
int eh_vazia (LISTA_ENC);
int tam (LISTA_ENC);
void ins (LISTA_ENC *, int, int);
int recup (LISTA_ENC, int);
void ret (LISTA_ENC *, int);
void destruir (LISTA_ENC);
```

UNIVASF

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



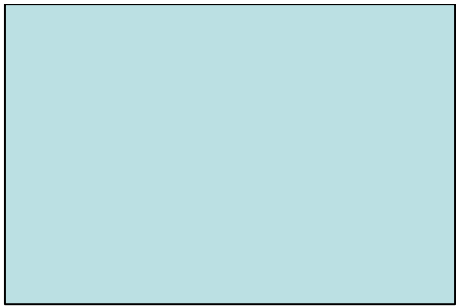
Alocação Encadeada

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

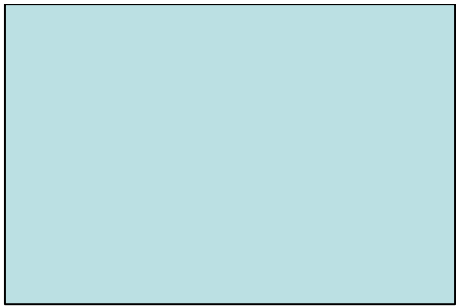
```
typedef struct nodo
{
   int inf;
   struct nodo * next;
}NODO;
typedef NODO * LISTA_ENC;
```

```
void cria_lista (LISTA_ENC *);
int eh_vazia (LISTA_ENC);
int tam (LISTA_ENC);
void ins (LISTA_ENC *, int, int);
int recup (LISTA_ENC, int);
void ret (LISTA_ENC *, int);
void destruir (LISTA_ENC);
```

```
void destruir (LISTA_ENC l)
{
    if (l)
    {
        destruir (l->next);
        free (l);
    }
}
```



Alocação Encadeada - Exercício



Implemente, no TAD LISTA_ENC, a seguinte operação:

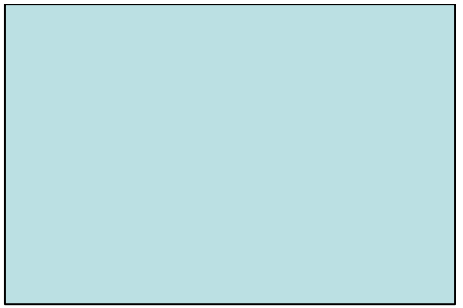
int pertence (LISTA_ENC l, int v)

a qual retorna 1 (um) se v pertence a lista l e 0 (zero) caso contrário.

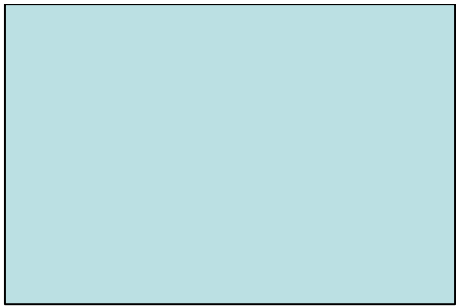
Faça duas implementações, uma sem fazer uso de recursividade e outra utilizando recursividade.

UNIVASF

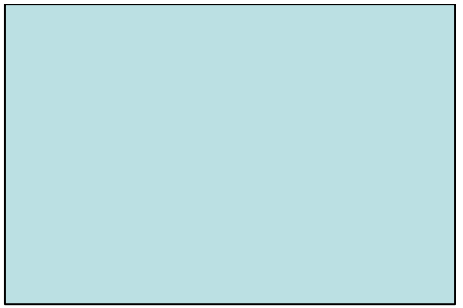
```
int pertence (LISTA_ENC l, int v)
{
   int t;
   for (t=tam(l); t>0; t--, l=l->next)
      if (l->inf==v)
         return (1);
   return (0);
}
```



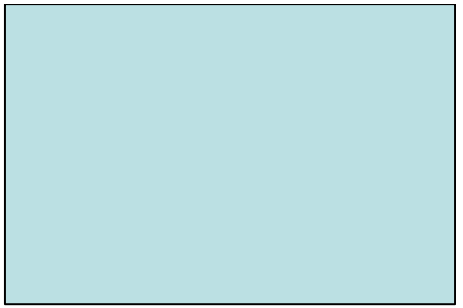
```
int pertence (LISTA_ENC l, int v)
{
   while (l)
   {
      if (l->inf==v)
         return (1);
      l=l->next;
   }
   return (0);
}
```



```
int pertence (LISTA_ENC l, int v)
{
   if (l)
   {
      if (l->inf==v)
         return (1);
      return (pertence (l->next, v));
   }
   return (0);
}
```



Alocação Encadeada - Exercício



Implemente, no TAD LISTA_ENC, a seguinte operação:

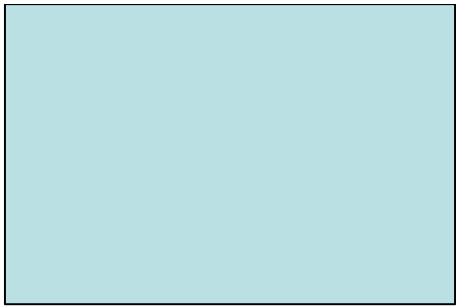
int eh_ord (LISTA_ENC l)

a qual retorna 1 (um) se a lista l está em ordem crescente e 0 (zero) caso contrário.

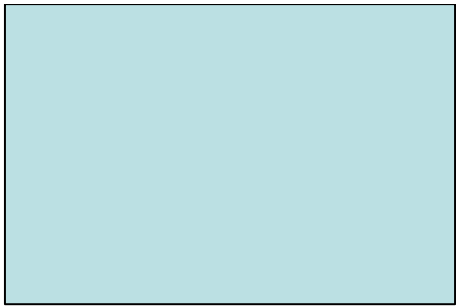
Faça duas implementações, uma sem fazer uso de recursividade e outra utilizando recursividade.

UNIVASF

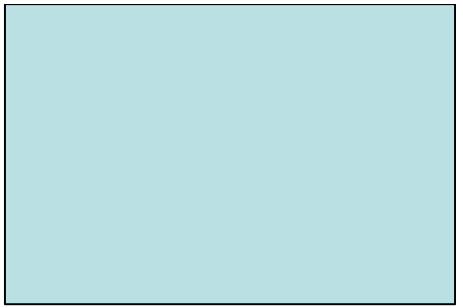
```
int eh_ord (LISTA_ENC l)
{
   int t;
   for (t=tam(l); t>1; t--, l=l->next)
      if (l->inf > (l->next)->inf)
         return (0);
   return (1);
}
```



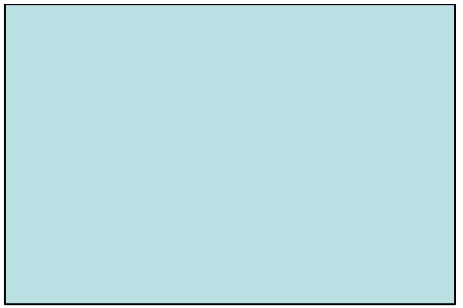
```
int eh_ord (LISTA_ENC l) {
   if (eh_vazia(l) || tam(l)==1)
      return (1);
   do
   {
      if (l->inf > (l->next)->inf)
         return (0);
      l=l->next;
   }while (l->next);
   return (1);
}
```



```
int eh_ord (LISTA_ENC l)
{
   if (!l || (l && !(l->next)))
      return (1);
   if (l->inf > (l->next)->inf)
      return (0);
   return (eh_ord (l->next));
}
```



Alocação Encadeada - Exercício



Implemente, no TAD LISTA_ENC, utilizando recursividade, a seguinte operação:

void gera_lista (LISTA_ENC *pl,int m,int n)

a qual utilizando-se das operações do TAD LISTA produz uma lista de inteiros correspondente a [m..n].