

```

void ins (LISTA_ENC_EST *l, int pos, float val) {
    if (pos<1 || pos> tam(l)+1) {
        printf("\nPosicao invalida!");
        exit (1);
    } else {
        if (l->ind_nodo_livre != -1) {
            int aux;
            if (pos==1) {
                l->elementos[l->ind_nodo_livre].inf=val;
                aux=l->elementos[l->ind_nodo_livre].next;
                l->elementos[l->ind_nodo_livre].next=l-
                >ind_pri_ele;
                l->ind_pri_ele=l->ind_nodo_livre;
                l->ind_nodo_livre=aux;
            }
        }
    }
}

```

```

else {
    int ind;
    for (ind=l->ind_pri_ele; --pos-1; ind=l-
>elementos[ind].next);
    l->elementos[l->ind_nodo_livre].inf=val;
    aux=l->elementos[l->ind_nodo_livre].next;
    l->elementos[l->ind_nodo_livre].next=l-
>elementos[ind].next;
    l->elementos[ind].next=l->ind_nodo_livre;
    l->ind_nodo_livre=aux;
}
} else {
    printf("\nImpossivel inserir novos
elementos.\nMemoria insuficiente.");
    exit (2); } } }

```

## Alocação Encadeada

Com base no que foi visto, implemente a operação que efetua a recuperação de um elemento na lista representada pelo TAD LISTA\_ENC\_EST.

**Dica:**

A posição é válida?

```

float recup (LISTA_ENC_EST *l, int pos)
{
    if (pos<1 || pos> tam(l))
    {
        printf("\nPosicao invalida!");
        exit (1);
    }
    else
    {
        int ind=l->ind_pri_ele;
        while (--pos)
            ind = l->elementos[ind].next;
        return l->elementos[ind].inf;
    }
}

```

## Alocação Encadeada

Com base no que foi visto, implemente a operação que efetua a remoção de um elemento na lista representada pelo TAD `LISTA_ENC_EST`.

### Dicas:

A posição é válida?

Todas as situações de remoção são tratadas da mesma forma?