```
//inciso parcial punteros

int *x  -- almacena la dirección de memoria de un entero
int A[] = { 1, 2, 3}
/// int A[3];
*(A+0)=1;
*(A+1)=2;
*(A+2)=3;

int b = 2; -> una variable entera que almacena un 2
int** y ->  puntero de (puntero a la dirección de un entero)
X= A + b -> A + 2 -> &A[2]
   Y = &x -> Almacena la direccion de X
       *(*(y)) = 4 -> *Y -> A+b -> &A[2] // *(&A[2]) ? A[2] = 4;


//esta función considera que la lista llega con datos.
int busquedaMinimo(nodo* lista)
{
    int minimo;
    if (lista->sig ¡= NULL)
    {
        mínimo = busquedaMinimo(lista->sig);
        if (lista->dato < mínimo)
        {
            return lista->dato;
        }
        else
        {
            return minimo;
        }
    }
    else
    {
        return lista->dato;
    }
}
min 1
2 -> 4 -> 6 -> 1 -> 3 -> NULL


//esta función considera que la lista llega con datos.
void buscarMinimo(nodo* lista, int* min)
{
    if (lista->sig != NULL)
    {
        buscarMinimo(lista->sig,min);
        if (lista->dato  < *min)
        {
            *min = lista->dato;
        }
    }
    Else
    {
        *min = lista->dato;
    }
}


void main()
{
    int min;
    int min2;
    Nodo* lista = NULL;
    cargaDatos(&lista);
    if (lista) // lista != NULL
```

```
    {
        buscarMinimo(lista,&min);
        printf(“El valor mínimo es %d”, min);
    }
///uso función con retorno para mínimo
    if (lista!=NULL)
    {
        min2= busquedaMinimo(lista);
        Printf(“El valor mínimo es %d”, min2);
    }
}
```