1.- ESPECIFICACIÓN:

tad ListaConPosicion(telemento) { Lista está formada por elementos de tipo telemento}

Especificación de las operaciones:

```
accion iniciarLista( sal  L : Lista )
{ Inicia L como una lista vacía }

accion insertar(e/s L:Lista, ent pos:entero, ent d:telemento)
{ Inserta d en la posición pos de L. Si la longitud de L es
menor que pos, lo añade al final }

funcion extraer( L : Lista, pos : entero ) devuelve telemento
{ Devuelve el elemento que ocupa la posición pos de L y no
modifica L }

accion eliminar(e/s  L : Lista, ent  pos:entero )
{ Modifica la lista L eliminando el elemento que ocupa la
posición pos }

funcion ListaVacia( L : Lista ) devuelve booleano
{ Si L está vacía devuelve VERDAD y FALSO en otro caso }

funcion longitud( L : Lista ) devuelve entero
{ Devuelve el número de elementos de la lista L }
```

IMPLEMENTACIÓN DINÁMICA:

- ***REPRESENTACIÓN:***

```
tipo
     Nodo = registro
          dato : telemento
          sig : puntero a Nodo
     freg;

     Lista = registro
          primero : puntero a Nodo
          long : entero
     freg;
```

- ***INTERPRETACIÓN:***

  Lista es un registro en cuyo campo long se almacena la longitud de la lista y cuyo puntero `primero` apunta a un nodo que contiene el elemento que ocupa la posición 1 en la lista y un puntero que apunta al nodo que contiene el elemento que ocupa la siguiente posición en la lista. El puntero del nodo correspondiente al elemento que ocupa la última posición de la lista apunta a `NULL`.

- ***IMPLEMENTACIÓN DE LAS OPERACIONES:***

```
accion iniciarLista( sal  L : Lista )
{ Inicia L como una lista vacía }

principio
     L.primero ← NULL
     L.long ← 0
fin



accion insertar( e/s  L : Lista, ent  pos:entero,
                 ent  d : telemento )
{ Inserta d en la posición pos de L. Si la longitud de
  L es menor que pos, lo añade al final }

variables
     aux, ant, nuevo : puntero a Nodo
     cont : entero
principio
     nuevo ← reservar(Nodo)
     si nuevo ≠ NULL entonces
          aux ← L.primero
          cont ← 1
          mientras que cont < pos AND aux ≠ NULL hacer
               cont ← cont +1
               ant ← aux
               aux ← dest(aux).sig
          fmq
          dest(nuevo).dato ← d
          dest(nuevo).sig ← aux
          si   aux = L.primero entonces
               L.primero ← nuevo
          si_no
               dest(ant).sig ← nuevo
          fsi
          L.long ← L.long + 1
     fsi
fin
```

```
funcion extraer( L : Lista, pos : entero ) devuelve
                                                    telemento
{ Devuelve el elemento que ocupa la posición pos de L
  y no modifica L }

variables
     aux : puntero a Nodo
     cont : entero
principio
     aux ← L.primero
     cont ← 1
     mientras que cont < pos hacer
          cont ← cont +1
          aux ← dest(aux).sig
     fmq
     devuelve(dest(aux).dato)
fin
```

```
accion eliminar(e/s  L : Lista, ent  pos:entero)
{ Modifica la lista L eliminando el elemento que ocupa la
  posición pos }

variables
     aux, ant: puntero a Nodo
     cont : entero
principio
     aux ← L.primero
     cont ← 1
     mientras que cont < pos hacer
          cont ← cont +1
          ant ← aux
          aux ← dest(aux).sig
     fmq
     si   aux = L.primero entonces
          L.primero ← dest(L.primero).sig
     si_no
          dest(ant).sig ← dest(aux).sig
     fsi
     liberar(aux)
     L.long ← L.long - 1
fin
```

```
funcion listaVacia( L : Lista ) devuelve booleano
{Si L está vacía devuelve VERDAD y FALSO en otro caso}

principio
     devuelve( L.long = 0 )
fin
```

```
funcion longitud( L : Lista ) devuelve entero
{ Devuelve el número de elementos de la lista L }

principio
     devuelve( L.long )
fin
```