ED – 20/10/2017

```
template <class T>
void Lista<T>:: Set(Posicion<T> p, const T&e){
	*p = e;
}

template <class T>
T Lista<T>:: Get(Posicion<T> p){
	return *p; //Tambien sería valido return *(p.i);
}


template <class T>
Posicion<T> Lista<T>:: Insertar(Posicion<T> p, const T&e){

//Aqui añadimos pos
	*int pos = p.i - (&(datos[0]));

	if(n == reservado)
		resize(2*reservados)

	p.i = &(datos[pos]);

	Posicion <T> q = end();

	aux = q;
	--aux;

	for(; q!=p; --q, --aux) -> No tiene inicializacion porque se ha hecho fuera (aux = q)
		*q = *aux;
	*q = e;
	n++;

	return q
}
```

Si hacemos int pos = p.i - (&(datos[0]));

En datos de 0 es donde empieza el vector, entonces al restar p.i - la pos de inicio, obtenemos la pos relativa de p.i (1,2,...)

```
template <class T>
```

```
Posicion <T> Lista<T>:: Borrar(Posicion<T> p){
        Posicion<T> siguiente = p;
        Posicion<T> q = p;

        for( ; siguiente != end(); ++q, ++siguiente)
                *q = *siguiente;
        n--;

        int pos = p.i-(&(datos[0]));

        if(n<reservados/4)
                resize(reservados/2);

        p-i = &(datos[pos]);

        return p;
}
```

- O -

LISTA: CELDAS ENLAZADAS CON UN UNICO PUNTERO

Nueva Forma:

```
Struct Celda{
        char ele;
        Celda *sig;
}

class Lista;

class Posicion{
private:
        Celda *punt;
public:
        Posicion(){
                punt(0);
                primera(0);
        }
        Posicion &operator++();
        Posicion &operator--(){

                Celda *p = primera;

                if(primera == punt)
                        return *this;
                else
```

```
                        while(p->sig != punt){
                                p = p->sig;
                        }
                        punt = P;
        }
        bool operator==(Posicion p);
        bool operator!=(Posicion p){
                return punt != p.punt;
        }
        char & operator*(){
                return punt->ele;
        }


        friend class Lista;
};


class Lista {
private:
        Celda *puntero;
        void Copiar(const Lista &L);
        void Borrar();
public:
        Lista();
        Lista(const Lista &L);
        ~Lista();
        Lista & operator=(const Lista & L);
        void Insertar(Posicion p, char e);
        void Borrar (Posicion p);
        char Get(Posicion p);
        void Set(Posicion p, char e);
        int size() const;
}
```