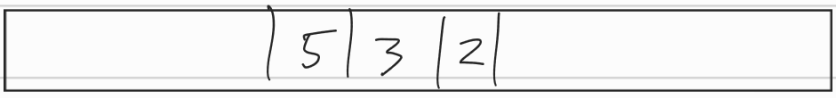
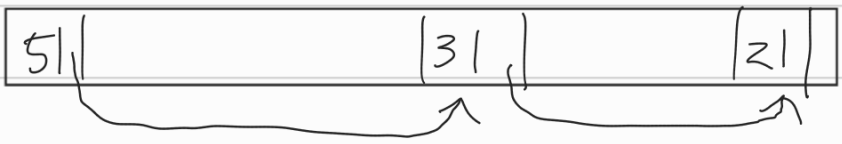
4. óra

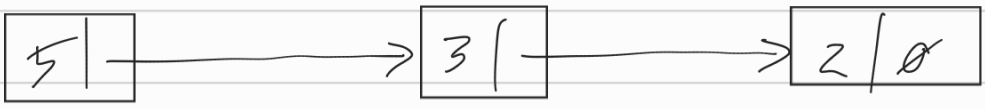
Memória:

- tömb esetén

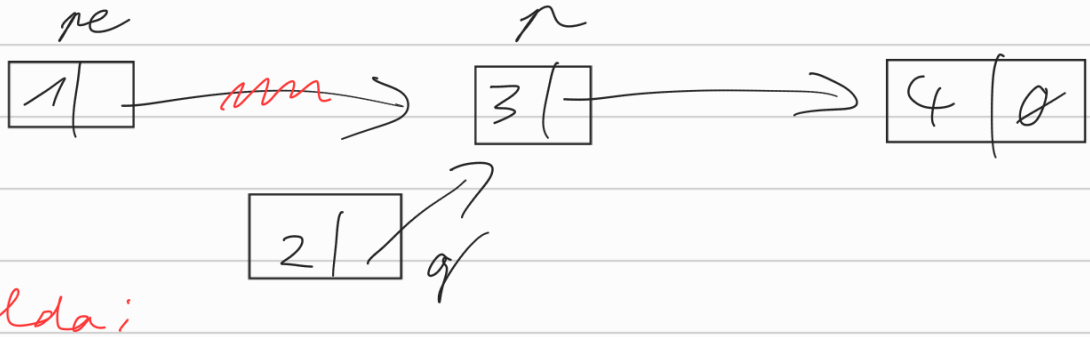


- láncolt lista esetén

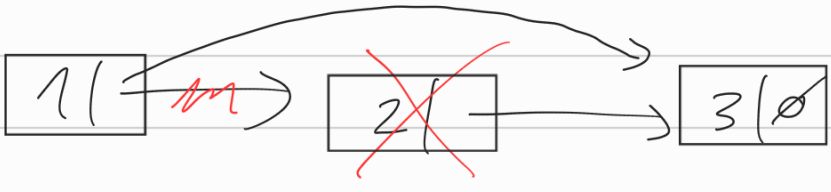




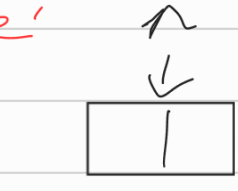
Beszúrásra példa:



Törlésre példa:



E1 típus:



key (Rules) next pointer

E1

p : E1*

p := new E1

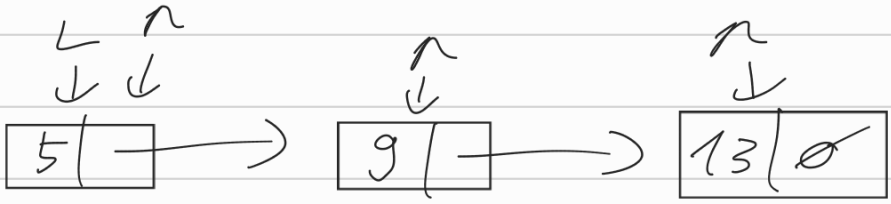
delete p

(*p).key

p -> key

Keresés:

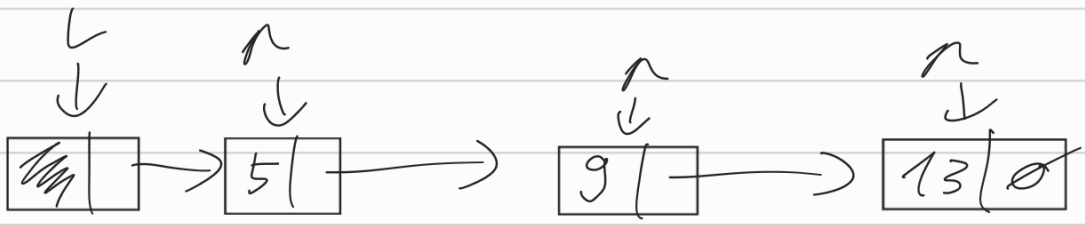
- SLL:



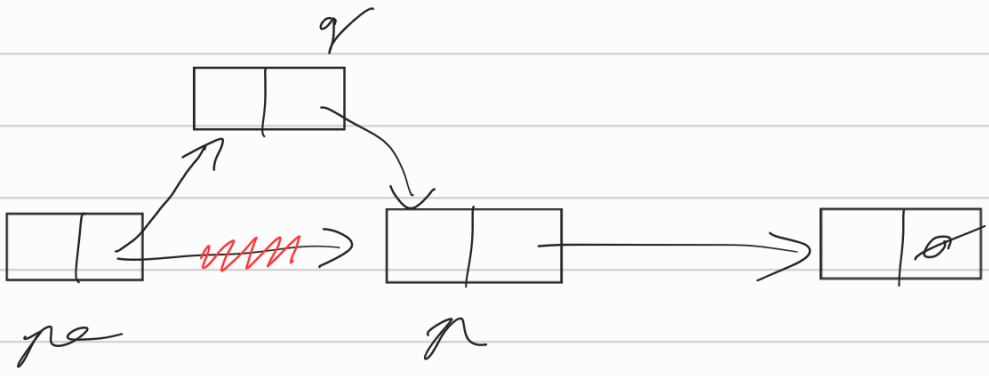
amíg p -> key < data

$p := p \rightarrow next$

- HL:

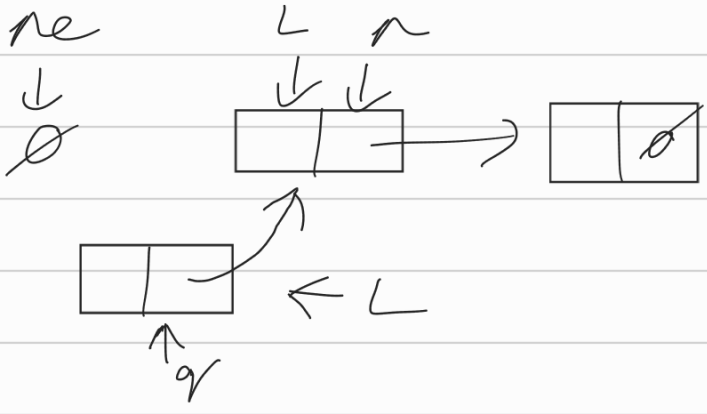


Beszúrás:

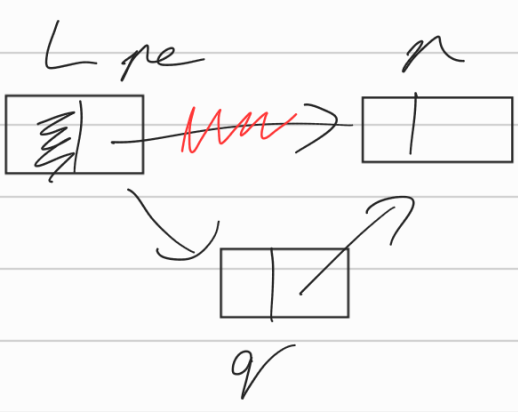


- ha az elejére kell beszúrni

SL

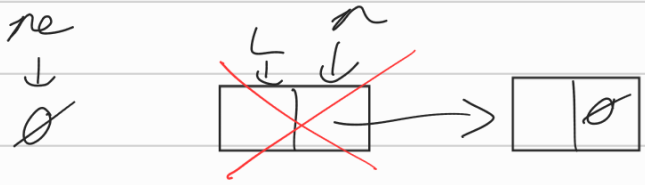


HL



Törlés:

- ha az elejéről kell SL esetben



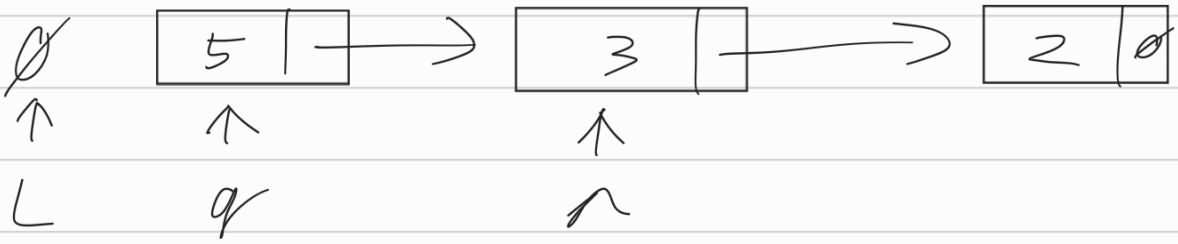
$L := p \rightarrow next$

- $pe \rightarrow next$ nem lehet

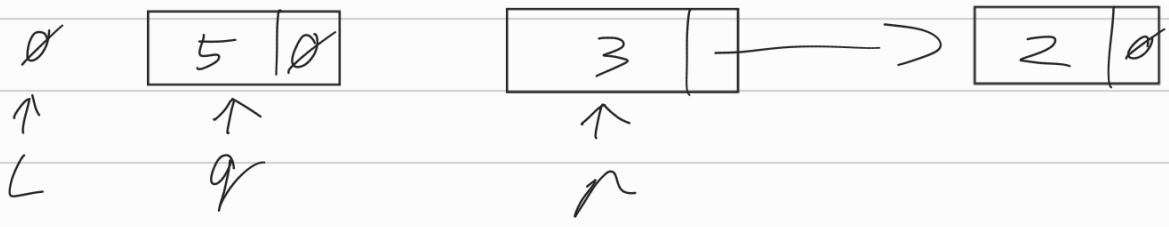
Lista megfordítása:

1.)

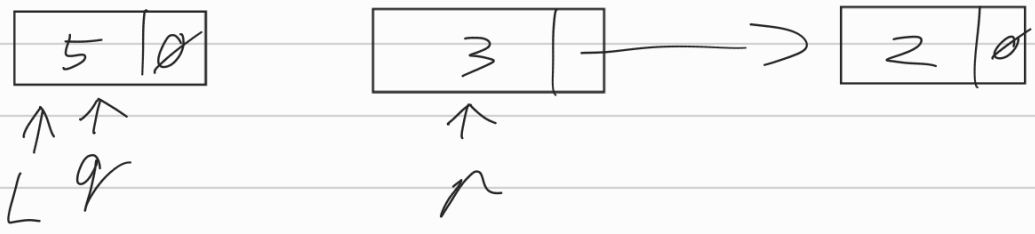
q := p



p := p -> next

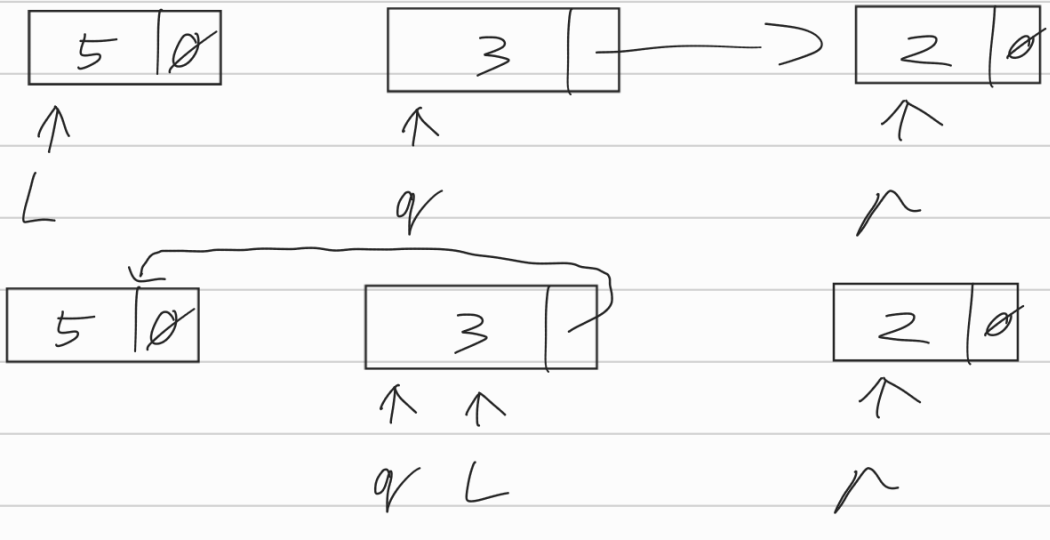


q -> next := L



L := q

2.)

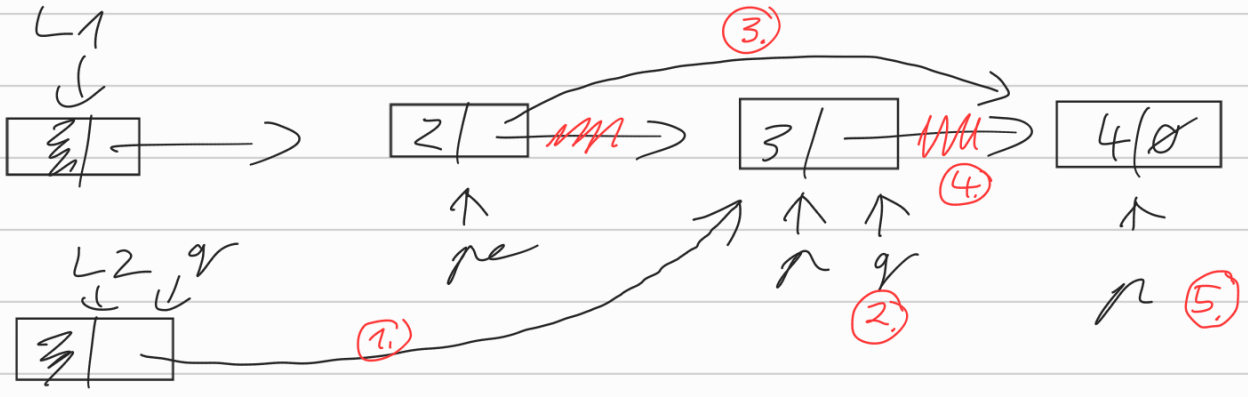


- L a megfordított lista lesz, ez kezdetben ∅
- a q az az elem, amit L-be befűzünk
- p azért kell, hogy ne vesszük el a listánkat

```
p := L
L := ∅
-----------
p != ∅
-----------
  q := p
  p := p -> next
  q -> next := L
  L := q
```

Lista szétfűzés:

- L1 a bemenet
- L1-ben legyenek a páros elemek
- L2-ben legyenek a páratlan elemek



1.) $q \to next := p$
2.) $q := p$
3.) $pe \to next := p \to next$
4.) $q \to next := \emptyset$
5.) $p := pe \to next$

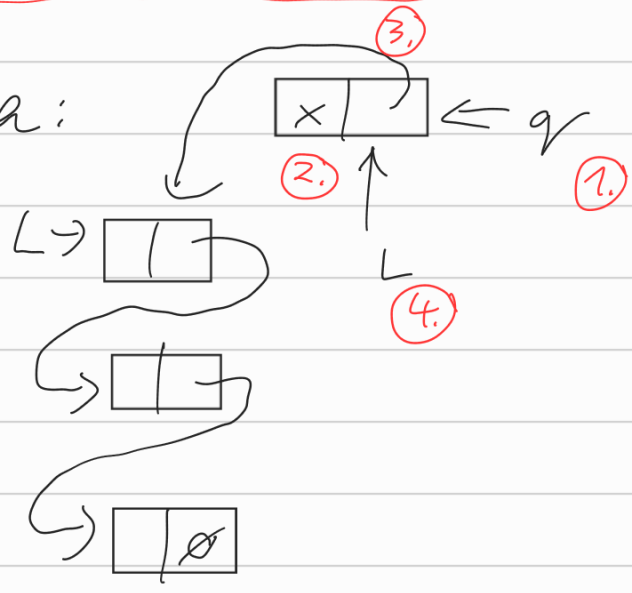
Prímszita:

- Húzzuk ki a számok többszöröseit
- listák esetén töröljük

2, 3, ~~4~~, 5, ~~6~~, 7, ~~8~~, ~~9~~, ~~10~~

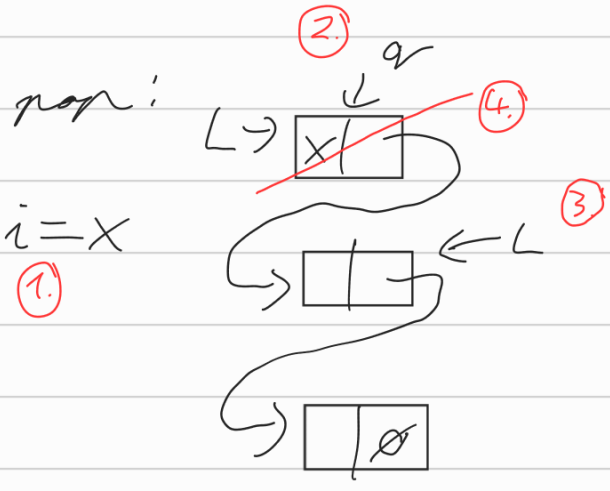
Verem láncoltan:

- push:




```
q := new E1
q -> key := X
q -> next := L
L := q
```

- pop:

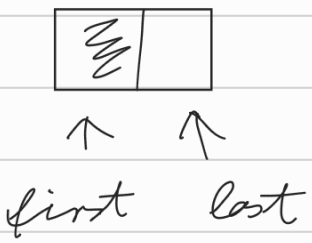


i = X
1.

```
i := L -> key
q := L
L := L -> next
delete q
return i
```

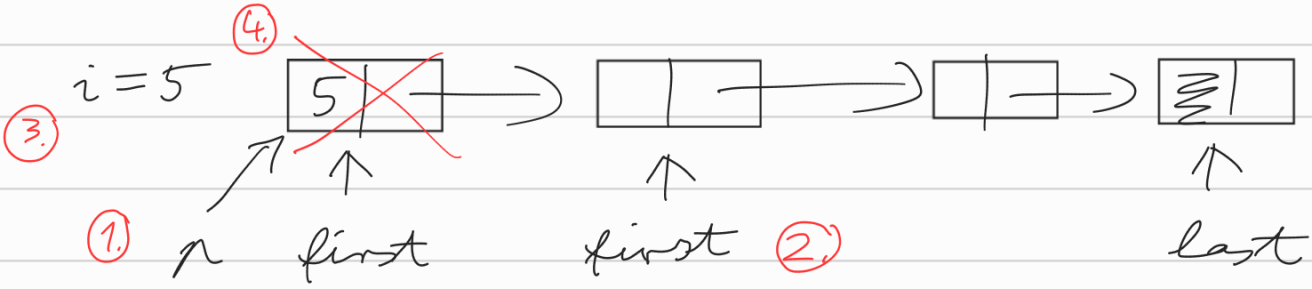
Sor láncoltan:

- kezdetben:



(végelem)

Sorból való törlés:




```
p := first
first := first -> next
i := p -> key
delete p
return i
```