Info2 TIN Liste chainée

① Insertion si chaine vide

```
bool insertElemAt(sChainedList *list, int32_t pos, sElem *x) {

  if(list==NULL || x==NULL) {
    return false;
  }

  if(list->numElem==0) {
    x->next = NULL;
    x->prev = NULL;
    list->first = x;
    list->last = x;
    list->numElem++;
  }
```

pos=0 élémt

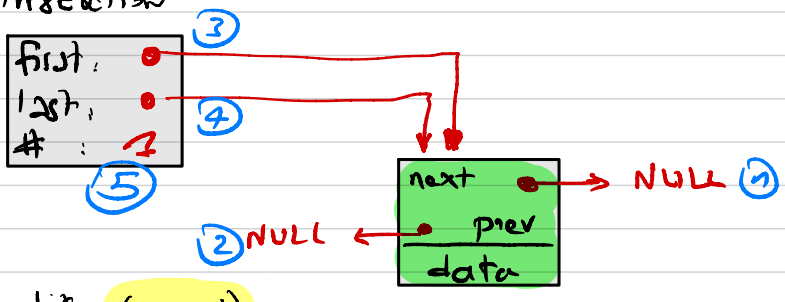
Valide les arguments

insertion dans une liste vide

AVANT INSERTION

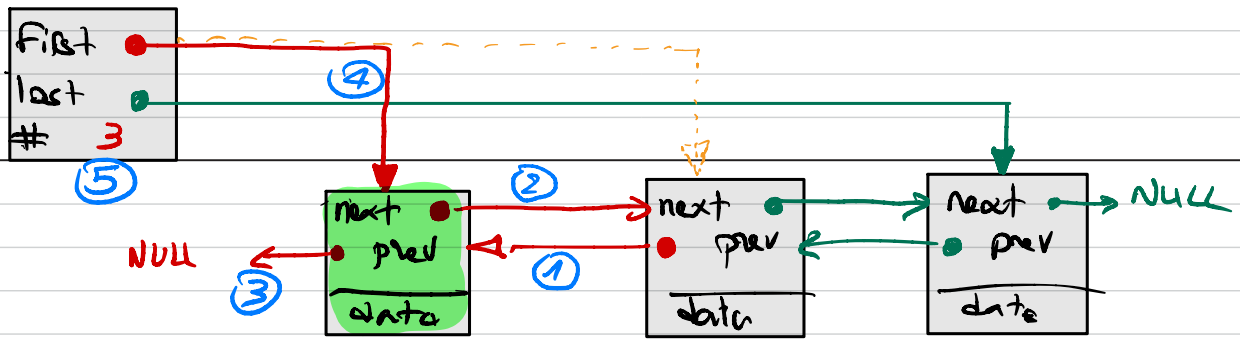
first: NULL
last: NULL
: 0

APRÈS INSERTION

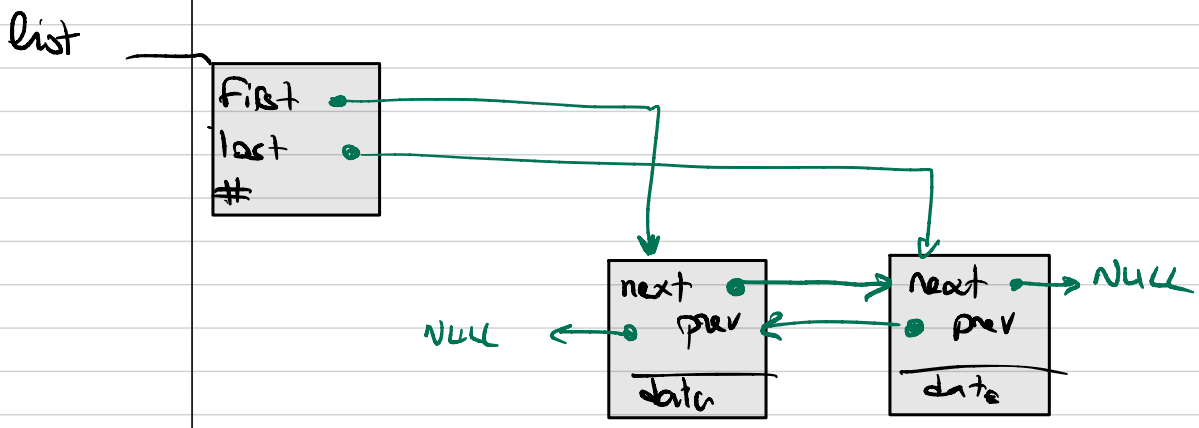


② Insertion en début de liste (pos=0)

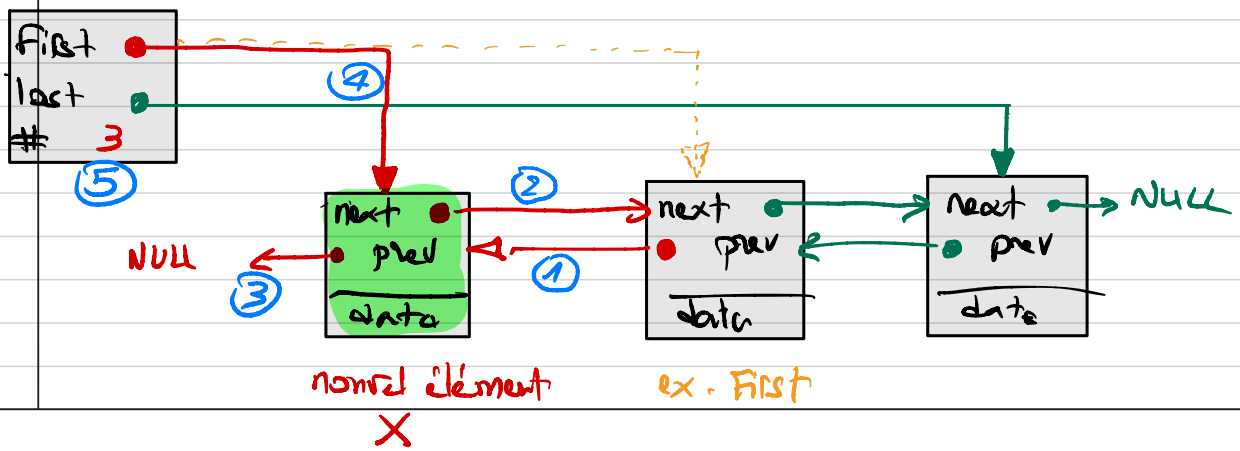
```
else if (pos == 0) {
  list->first->prev = x;
  x->next = list->first;
  x->prev = NULL;
  list->first = x;
  list->numElem++;
}
```



Condition de départ

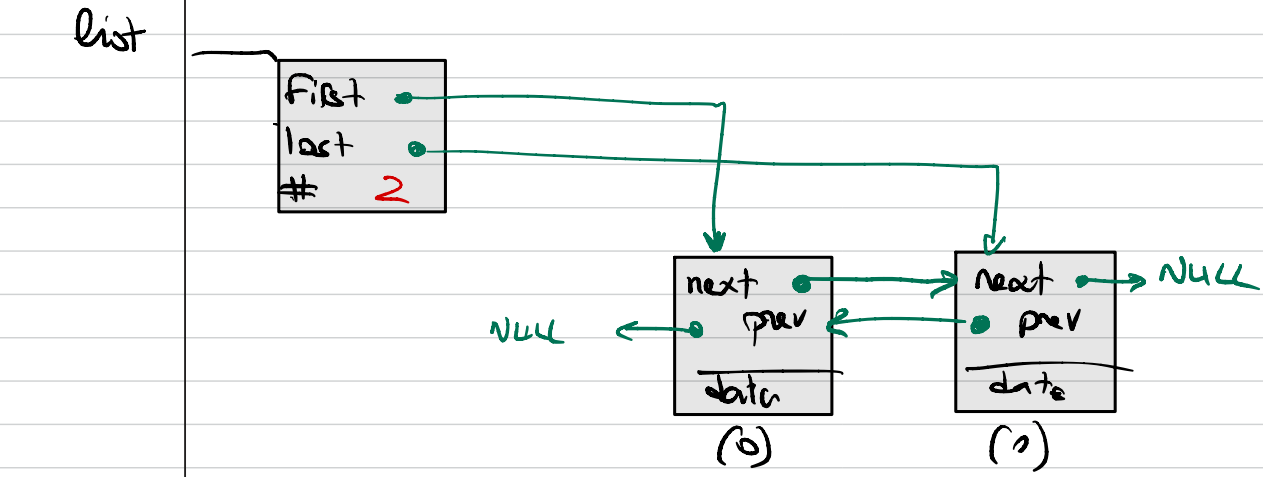


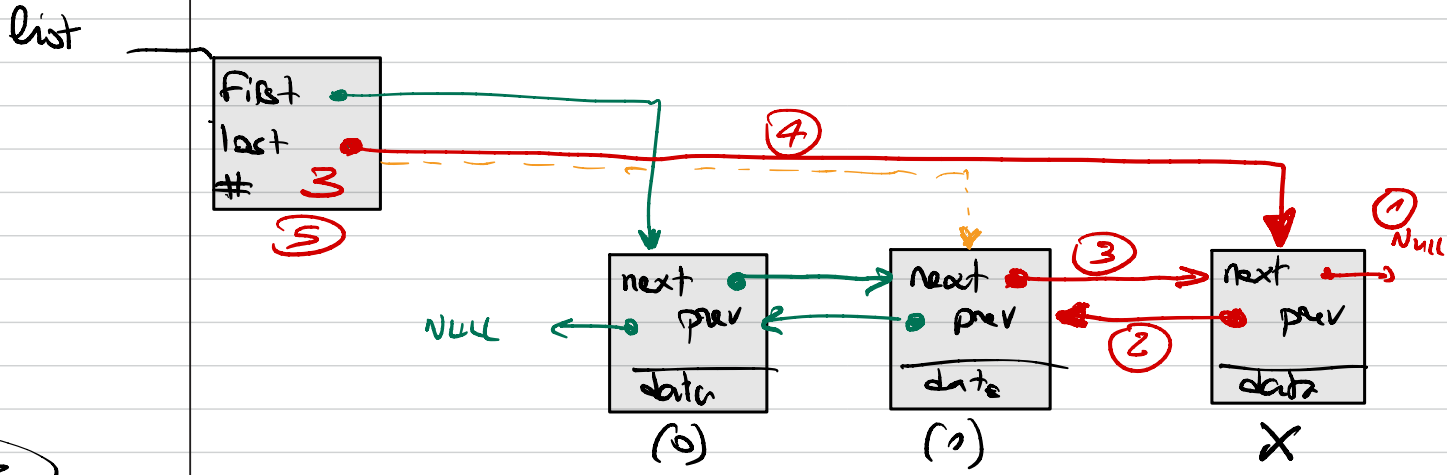

```
else if (pos == 0) {
  list->first->prev = x;
  x->next = list->first;
  x->prev = NULL;
  list->first = x;
  list->numElem++;
}
```



END_OF_LIST

Insertion en fin de liste (pos = -1 ou pos = nbre élément)



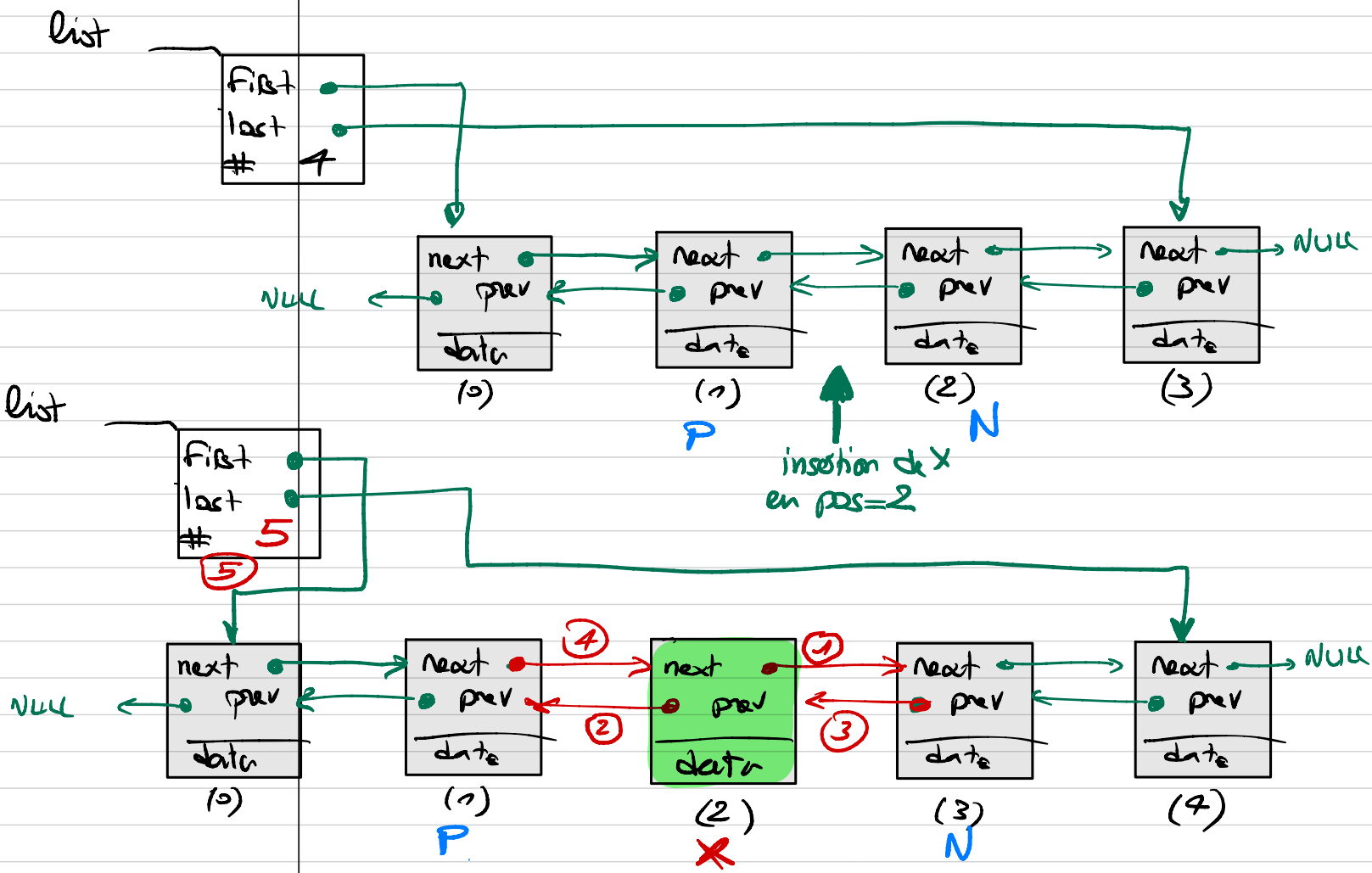


16 h 38

```
else   if(pos==END_OF_LIST || pos==list->numElem)
  x->next = NULL; (1)
  x->prev = list->last; (2)
  list->last->next = x; (3)
  list->last = x; (4)
  list->numElem++; (5)
}
```

Test pos = fin.

Insertion en milieu de Liste



1) n : list → first
2) décaler n = n → next ↺ nbre d'itérations = pos

p = n → prev

17h05

```
else {
  n = list->first;
  for (k = 0; k < pos;k++) {
    n = n->next;
    p = n->prev;
  }
  x->next = n;  (1)
  x->prev = p;  (2)
  n->prev = x;  (3)
  p->next = x;  (4)
  list->numElem++;  (5)
}
```

→ Recherche de n et p.

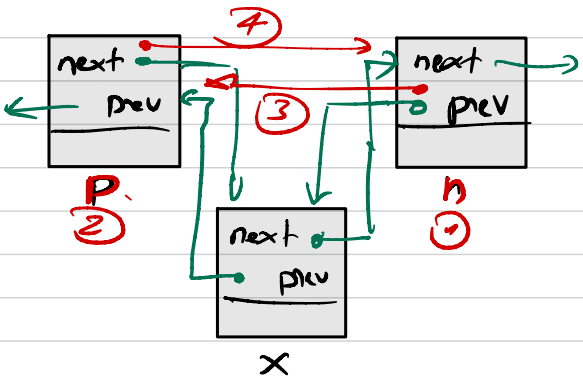
```
sElem *p = NULL;
sElem *n = NULL;
int32_t k = 0;
```

Suppression d'un élément

a) 1er (pos = 0)
b) dernier (pos = -1 ou pos = list → numElem - 1)
c) à une autre position.

On utilise le même cheminement que précédemment.

c) faire pointer x sur l'élément "pos" (for)
mémoriser x
1. n = pos → next
2. p = pos → prev
3. n → prev = p
4. p → next = n
5. list → numElem --



⚠ Il faut pouvoir "libérer" la mémoire occupée par x.

Parcours pour la liste ⇄ recherche, insertion à une position

Sens → n = list → first

Pour passer à l'élément suivant : n = n → next
On peut faire ça tant que n ≠ NULL

Sens ← n = list → last

Pour passer à l'élément précédent : n = n → prev
On peut faire ça tant que n ≠ NULL

Solution à venir : - Suppression
- Affichage Liste → ou ←

Bonnes Révisions à Tous !