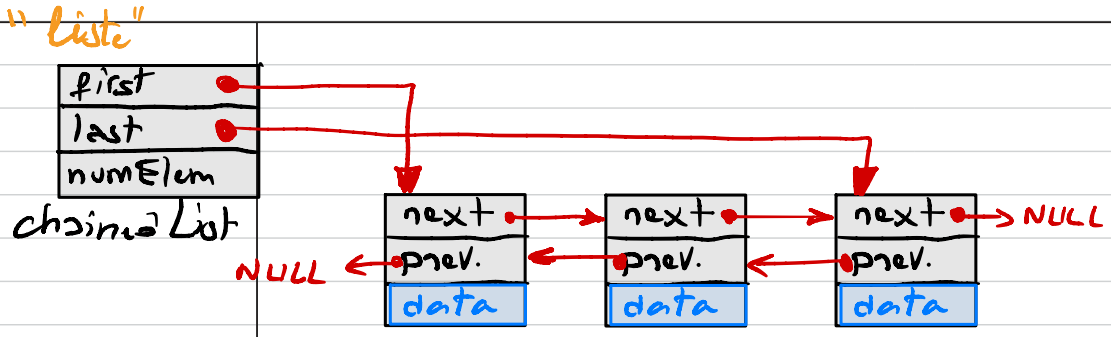
Liste Chaînée (suite)

10.[illegible].2021




```c
typedef struct sElem {
  struct sElem *next;
  struct sElem *prev;
  Element e;
} sElem;

typedef struct {
  sElem *first;
  sElem *last;
  uint32_t numElem;
} chainedList;
```

main chainedList list = { NULL, NULL, 0 };
(first) (last) (numElem)

sElem *p;

p = createElem(data);

insertElemAt(&list, data, pos);

TD2021610 :
- TD2021610.c avec dans le main.
- chainedList.c / .h
  - ↳ fonction createElem

Element (data) → createElem → sElem *

16h52

```c
// chained_list.h

#pragma once

#include <stdlib.h>
#include <stdint.h>

typedef double Element; // element in list (payload)

typedef struct sElem {
  struct sElem *next;
  struct sElem *prev;
  Element e;
} sElem;

typedef struct {
  sElem *first;
  sElem *last;
  uint32_t numElem;
} chainedList;

sElem *createElem(Element data);
```

structure pour l'élément

structure de la liste

prototype de création d'un élément

```c
// chained_list.c

#include "chained_list.h"


sElem *createElem(Element data) {

  sElem *p = (sElem *)malloc(sizeof(sElem));
  if(NULL!=p) {
    p->e = data;
    p->next = NULL;
    p->prev = NULL;
  }
  return p;
}
```

allocation de l'élément

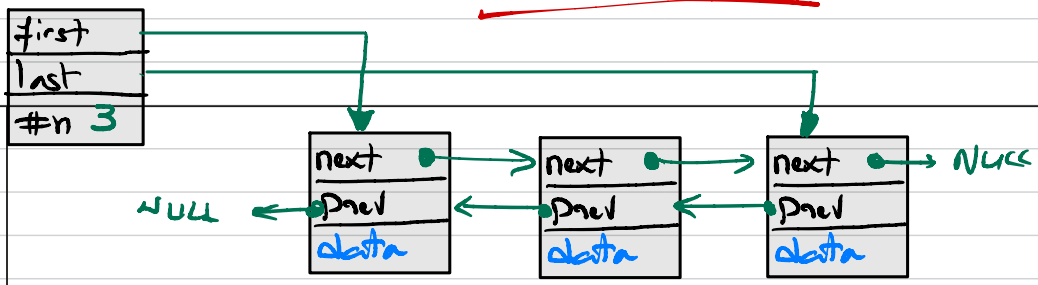
MAJ de la payload

init des pointeurs pour la liste

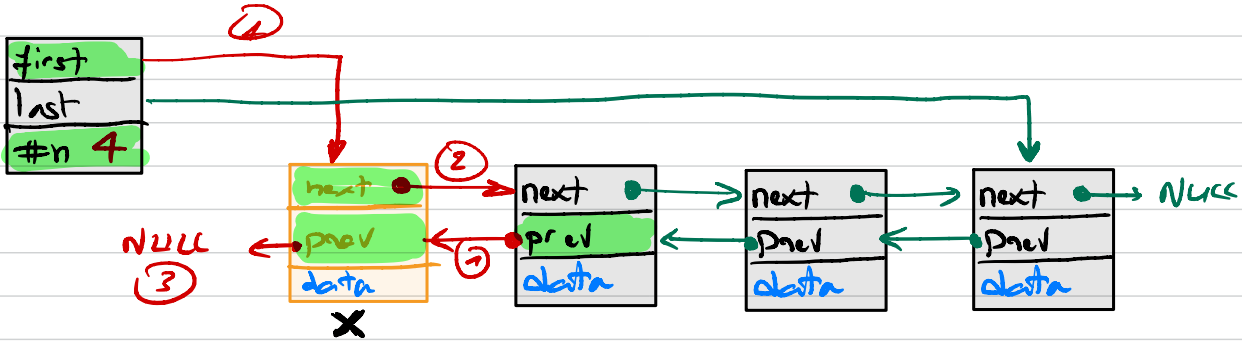
Insertion en début de liste.

pos = ∅

AVANT



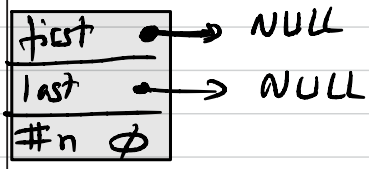
APRÈS



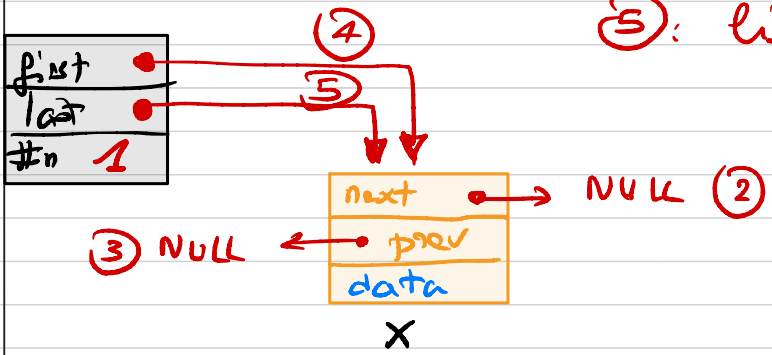
① : list → first → prev = X
② : X → next = list → first
③ : X → prev = NULL
④ : list → first = X

⚠ Si liste vide : list → first = NULL
list → numElem == ∅ .
l'opération ① n'est pas réalisable

AVANT



APRÈS



⑤ : list → last = X

list → numElem ++ Dans tous les cas.

insertElemAt (list , elem , pos) ;
∅

17h25

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

  if(pos==0){ // insert at first position
    if(list->numElem==0) { // list is empty
      x->next = NULL;
      x->prev = NULL;
      list->first = x;
      list->last = x;
    }
    else {
      list->first->prev = x;
      x->next = list->first;
      x->prev = NULL;
      list->first = x;
    }
    list->numElem++;
  }
  return true;
}
```

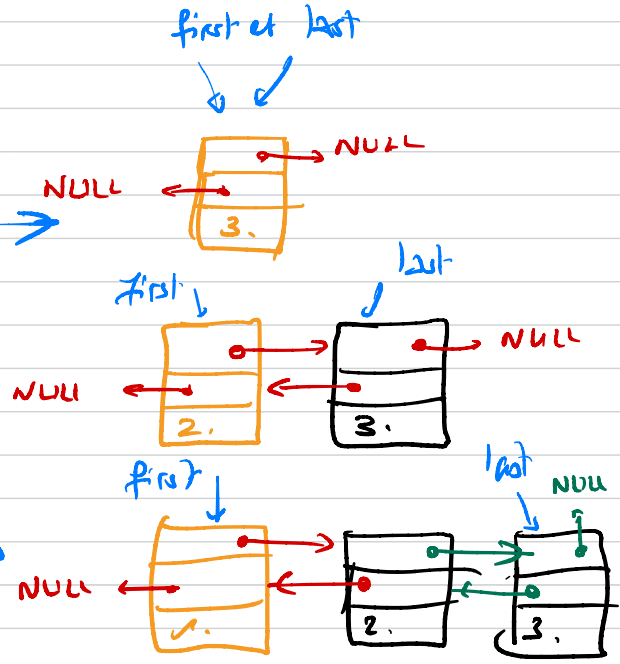
(au début)

x : seul élément de la liste.

// list not empty

← M.A.J. # nb elem

```c
int main(int argc, char const *argv[])
{
  sElem *elem;
  chainedList l = {NULL, NULL, 0};

  elem = createElem(3.);
  printf("%lf\n", elem->e);

  insertElemAt(&l, 0, elem);

  elem = createElem(2.);
  insertElemAt(&l, 0, elem);

  elem = createElem(1.);
  insertElemAt(&l, 0, elem);

  displayList(&l);
```



1. 2. 3.

```c
void displayList(chainedList *l) {
  sElem *e = l->first;
  printf("N=%d\n", l->numElem);
  while(e) { // e != NULL
    printf("%lf\n", e->e);
    e = e->next;
  }
  return;
}
```

1er élément

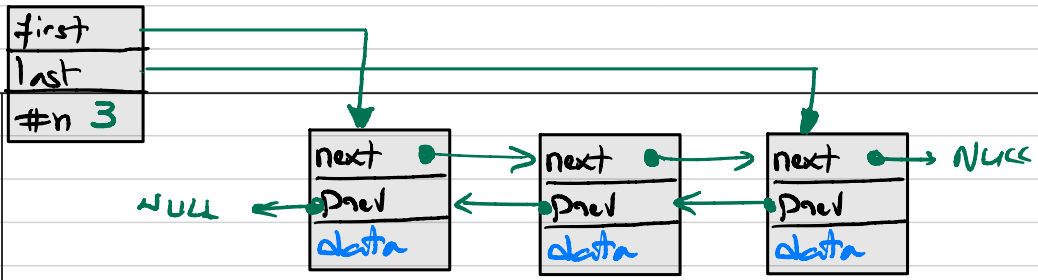
nbre élément

on passe au suivant

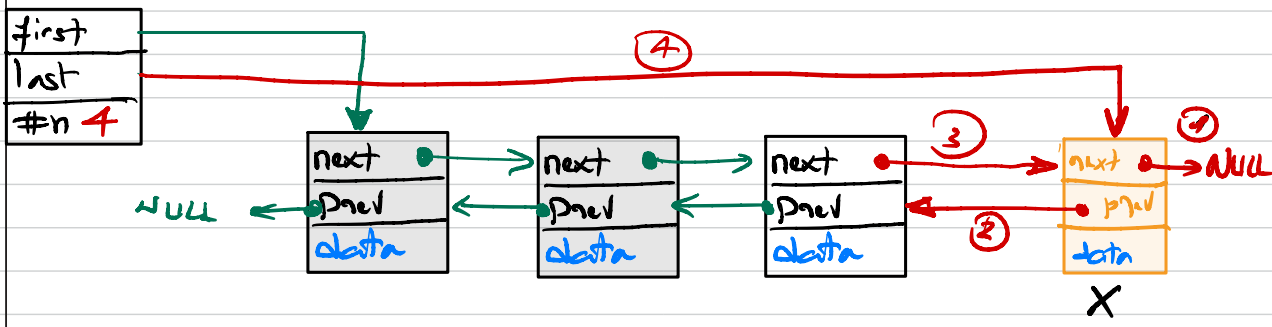
Insertion en fin de liste

POS = -1 (END_OF_LIST)

AVANT



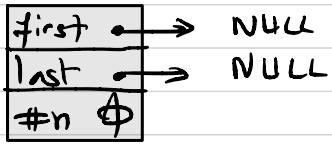
APRÈS



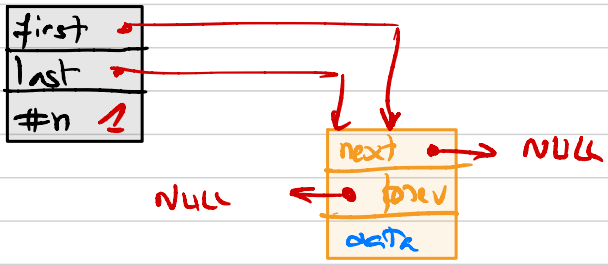
1. X → next = NULL
2. X → prev = list → last
3. list → last → next = X
4. list → last = X

① Liste vide

AVANT



APRÈS



idem cas avec POS = 0.

list → numElem++

dans tous les cas.

17h32

Suite de la fonction insertElemAt :

```
else   if(pos==END_OF_LIST || pos==list->numElem) { // insert at last position
  if(list->numElem==0) { // list is empty
    x->next = NULL;
    x->prev = NULL;
    list->first = x;
    list->last = x;
  }
  else {
    x->next = NULL;
    x->prev = list->last;
    list->last->next = x;
    list->last = x;
  }
  list->numElem++;
}
```

cas d'insertion à la fin

si liste vide

```
int main(int argc, char const *argv[])
{
  sElem *elem;
  chainedList l = {NULL, NULL, 0};

  elem = createElem(3.);
  printf("%lf\n", elem->e);
  insertElemAt(&l, 0, elem);

  elem = createElem(2.);
  insertElemAt(&l, 0, elem);

  elem = createElem(1.);
  insertElemAt(&l, 0, elem);

  displayList(&l);

  elem = createElem(4.);
  insertElemAt(&l, END_OF_LIST, elem);

  elem = createElem(5.);
  insertElemAt(&l, END_OF_LIST, elem);

  elem = createElem(6.);
  insertElemAt(&l, 5, elem);


  displayList(&l);
```

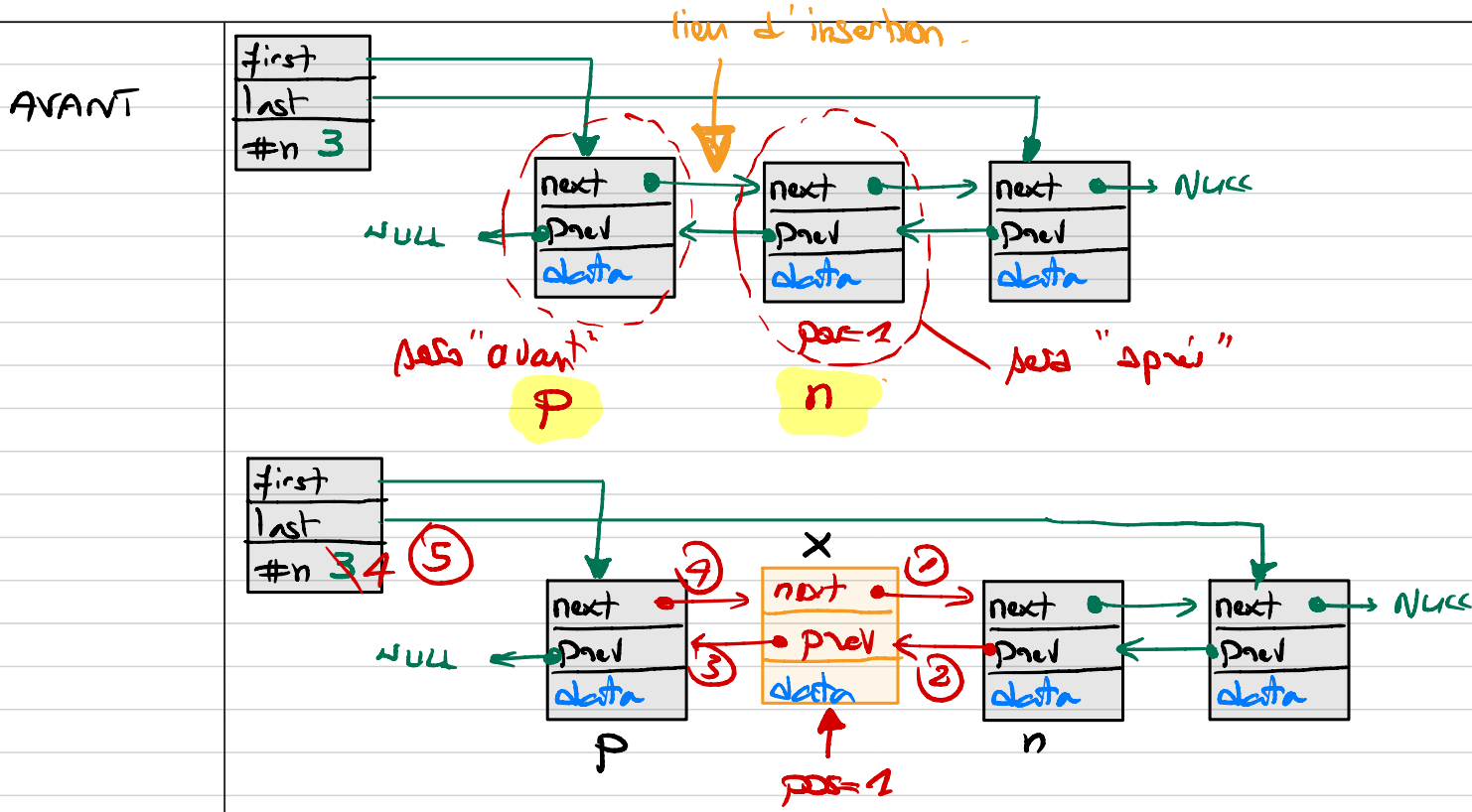
insertion en début

insertion en fin de liste

(-1)

dernière position

Insertion "dans" la liste | 0 < pos < numElem



la position d'insertion = position où est le nouvel élément après avoir réalisé l'insertion

① identifier "p" et "n"

Se placer au début de la liste.
Se déplacer d'élément en élément "pos fois"

ex: pos = 1.

```
n = list -> first
for(k = 0; k < pos; k++)
    n = n -> next.        <- n est positionné
p = n -> prev;
```

① x -> next = n
② n -> prev = x
③ x -> prev = p
④ p -> next = x
⑤ list -> numElem ++

- Implémentez cette insertion dans le TD + tôt dans le main.
- displayListReverse (du dernier jusqu'au premier).