INSTITUTO FEDERAL
SÃO PAULO
Campus Guarulhos

Lista Ligada
(Lista Encadeada)

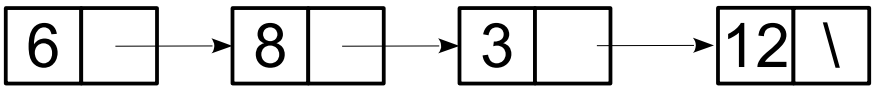
Prof. Gláucya Boechat
gcbcht@gmail.com

INSTITUTO FEDERAL
SÃO PAULO
Campus Guarulhos

Lista Ligada

- Estrutura de dados linear ou dinâmica.

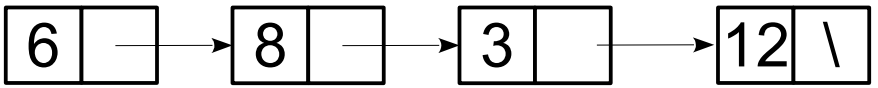
- A lista encadeada ou lista ligada é uma sequencia de elementos (nós), onde cada nó possui:
  - Um ou mais campos de informações
  - Um ponteiro para o próximo nó da lista



INSTITUTO FEDERAL
SÃO PAULO
Campus Guarulhos

Lista Ligada

- Exemplo
  - Dinâmica



  - Linear

6	8	3	12

INSTITUTO FEDERAL
SÃO PAULO
Campus Guarulhos

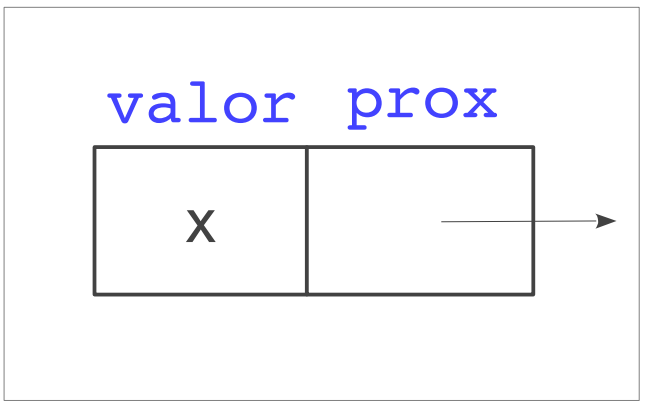
Lista

- Conjunto de operações
  - Inicializar lista vazia
  - Inserir um elemento após o i-ésimo elemento da lista
  - Alterar o i-ésimo elemento da lista
  - Ordenar a lista em ordem crescente ou decrescente
  - Remover o i-ésimo elemento da lista
  - Concatenar uma lista em outra lista.

Lista Encadeada

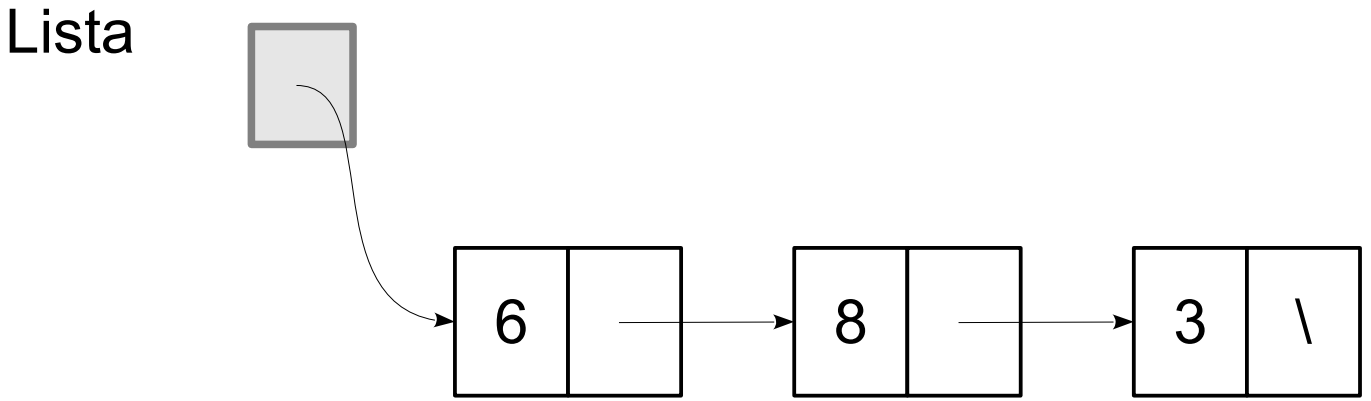
- lista.c

```
#include <stdio.h>
#include <stdlib.h>


typedef struct no {
   int valor;
   struct no* prox;
}No;
```



INSTITUTO FEDERAL
SÃO PAULO
Campus Guarulhos

Lista Encadeada



INSTITUTO FEDERAL
SÃO PAULO
Campus Guarulhos

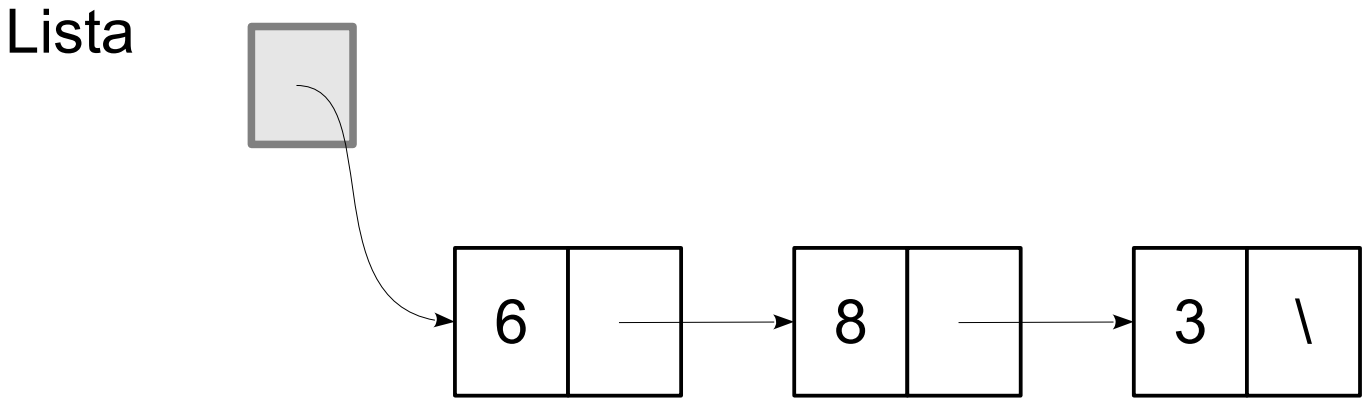
Lista Encadeada

- lista.c

```
void imprime_lista(No* lista) {
  while (lista != NULL) {
    printf("%d\n", lista->valor);
    lista = lista->prox;
  }
}
```

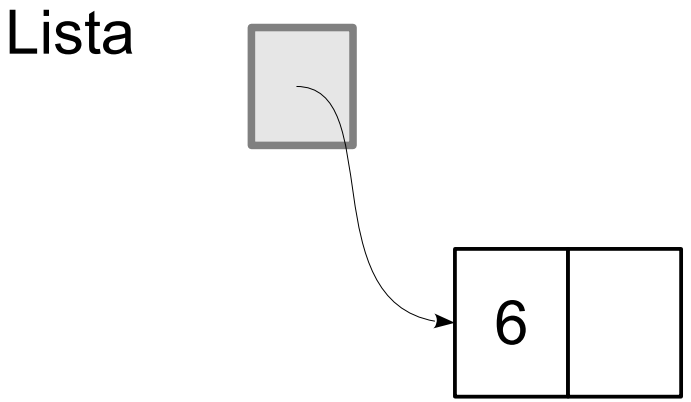
INSTITUTO FEDERAL
SÃO PAULO
Campus Guarulhos

Lista Encadeada



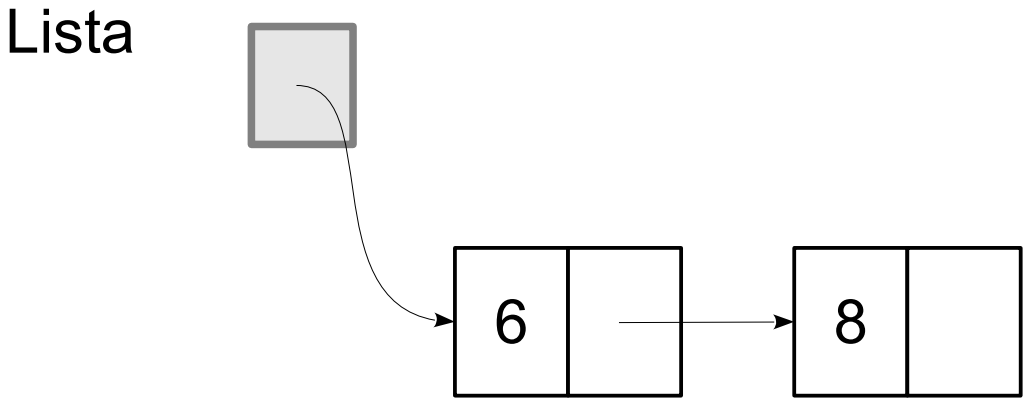
INSTITUTO FEDERAL
SÃO PAULO
Campus Guarulhos

Lista Encadeada



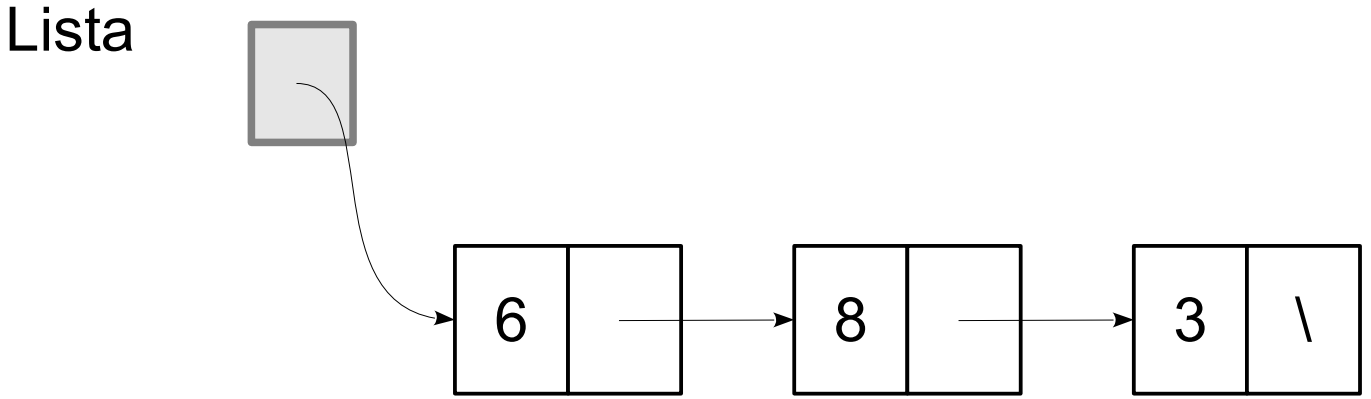
INSTITUTO FEDERAL
SÃO PAULO
Campus Guarulhos

Lista Encadeada



INSTITUTO FEDERAL
SÃO PAULO
Campus Guarulhos

Lista Encadeada



Lista Encadeada

- Função

```
void insere_comeco(No** lista, int valor{
   No* no;
   no = (No*) malloc(sizeof(No));

   no->valor = valor;
   no->prox = *lista;

   *lista = no;

   imprime_lista(*lista);
}

//   ...
```

INSTITUTO FEDERAL
SÃO PAULO
Campus Guarulhos

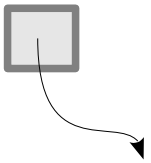
Lista Encadeada

```
void main(){

   No* lista;

   lista = NULL;

   insere_comeco(&lista, 3);
   insere_comeco(&lista, 8);
   insere_comeco(&lista, 6);
   insere_comeco(&lista, 6);

   imprime_lista(lista);
}
```

INSTITUTO FEDERAL
SÃO PAULO
Campus Guarulhos

Inicializar a Lista

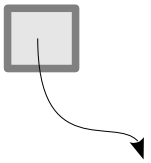
X = 3

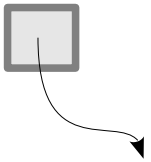
Lista 

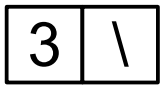
INSTITUTO FEDERAL
SÃO PAULO
Campus Guarulhos

Inicializar a Lista

X = 3

Lista 

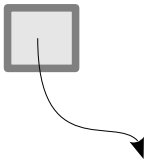
Lista 

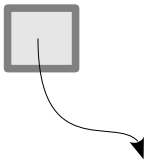


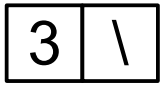
INSTITUTO FEDERAL
SÃO PAULO
Campus Guarulhos

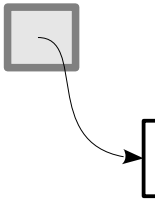
Inicializar a Lista

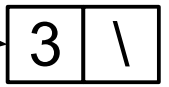
X = 3

Lista 

Lista 



Lista 

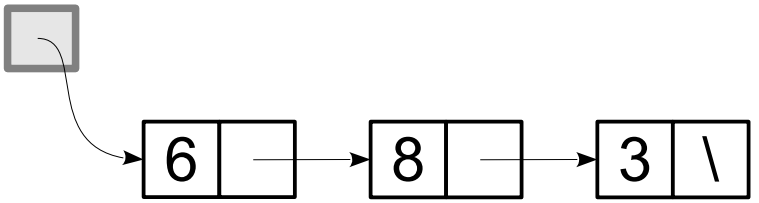


INSTITUTO FEDERAL
SÃO PAULO
Campus Guarulhos

Inserir no início da lista

X = 10

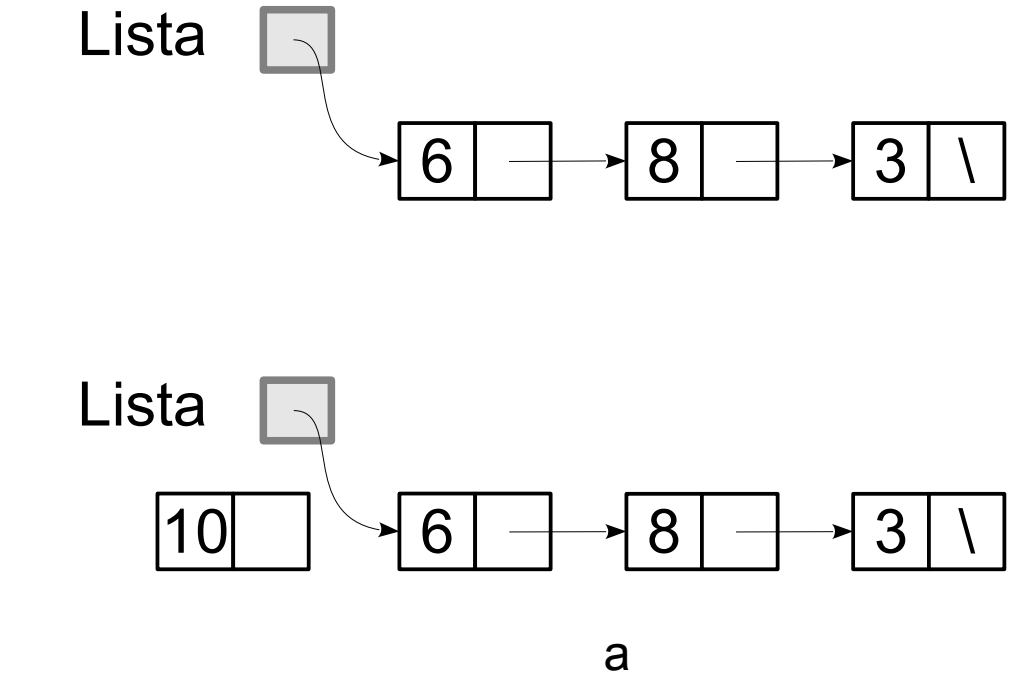
Lista



INSTITUTO FEDERAL
SÃO PAULO
Campus Guarulhos

Inserir no início da lista

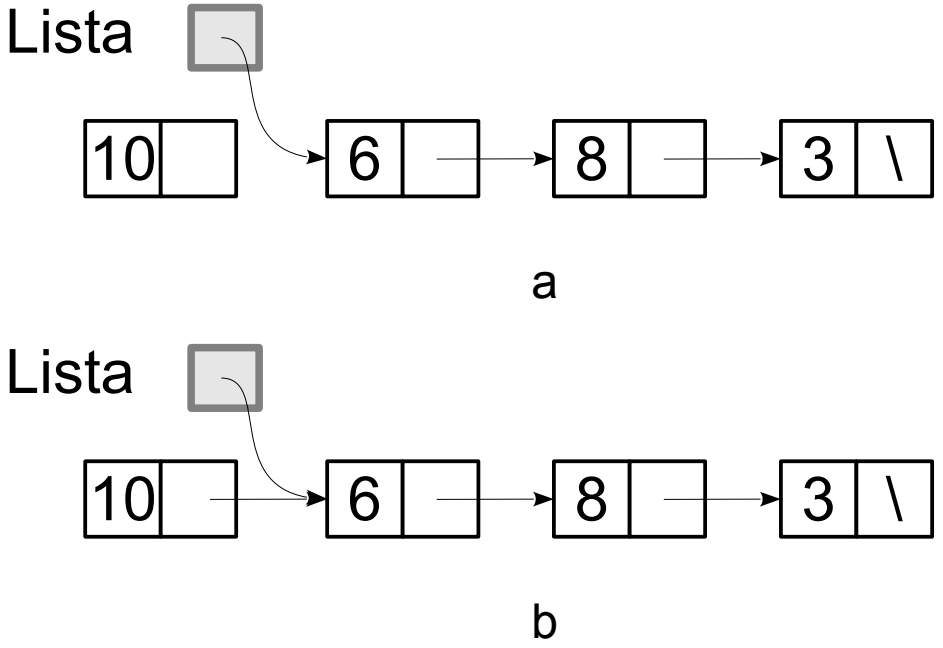
X = 10



a

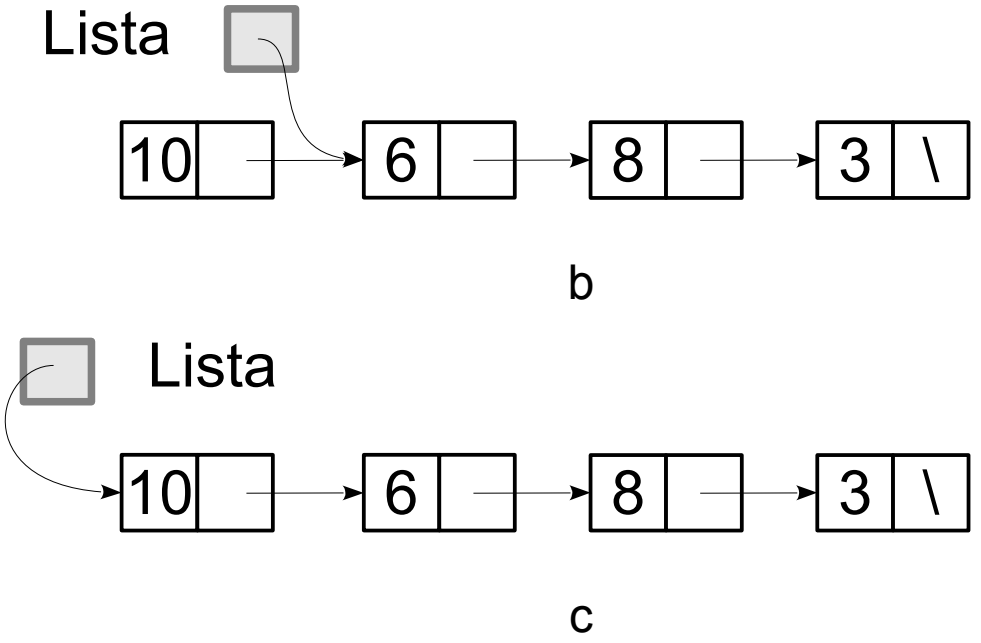
INSTITUTO FEDERAL
SÃO PAULO
Campus Guarulhos

Inserir no início da lista



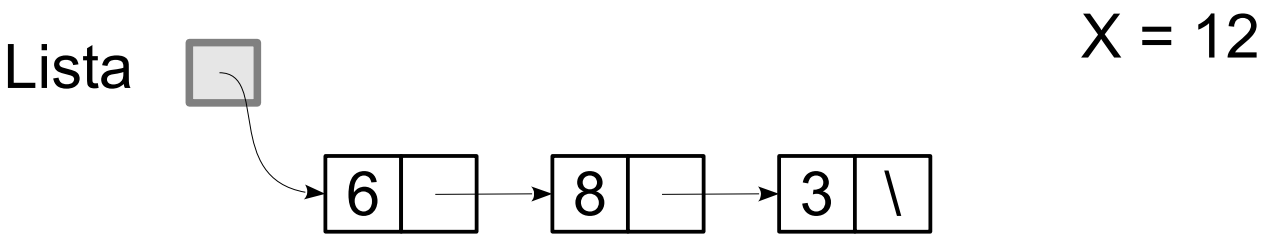
INSTITUTO FEDERAL
SÃO PAULO
Campus Guarulhos

Inserir no início da lista



INSTITUTO FEDERAL
SÃO PAULO
Campus Guarulhos

Inserir no final da lista



Inserir no final da lista

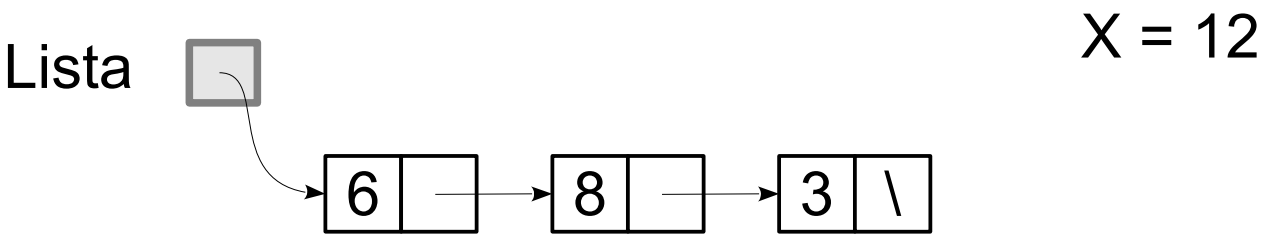
```
void insere_final(No** lista, int v) {

   No *n, *aux;
   n = (No*) malloc(sizeof(No));
   n->valor = v;
   n->prox = NULL;  // Ultimo elemento da lista

   if (*lista == NULL)
      *lista = n;
   else {
      aux = *lista;
      while (aux->prox != NULL)
         aux = aux->prox;
      aux->prox = n;
   }
}
```

Inserir no final da lista

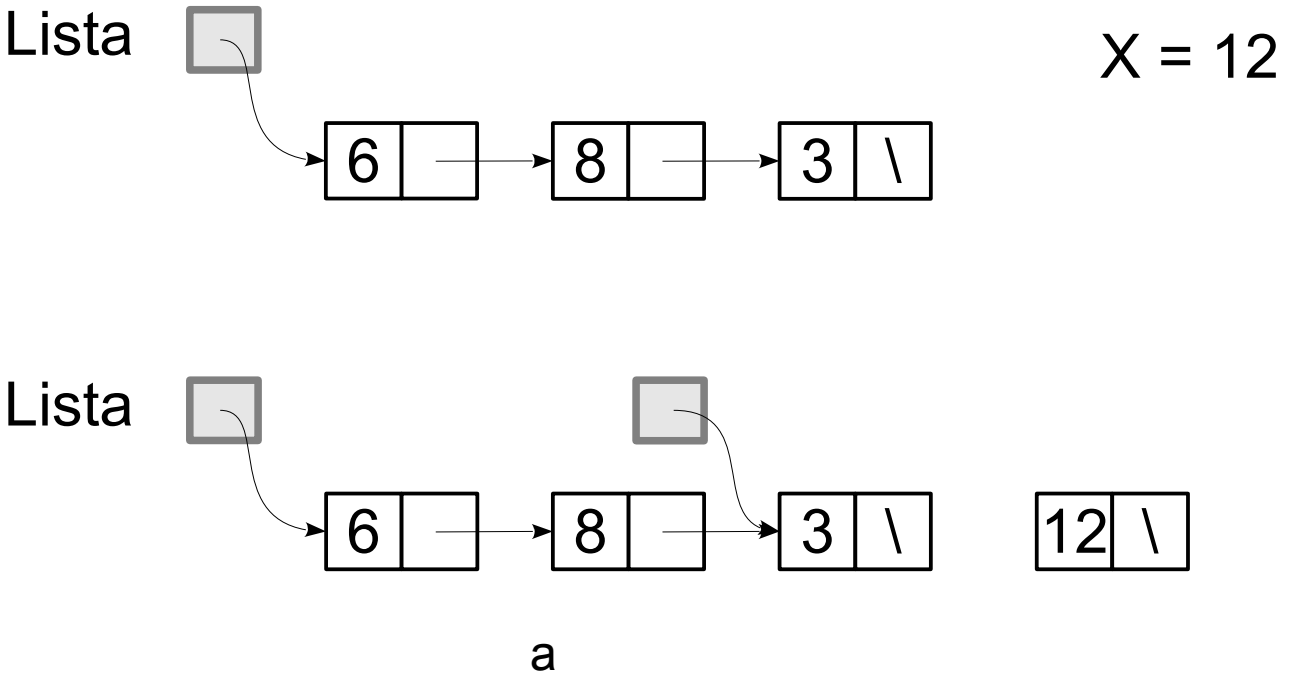
```
void main() {

   ...
   insere_final(&lista, 12);
   imprime_lista(lista);

}
```

INSTITUTO FEDERAL
SÃO PAULO
Campus Guarulhos

Inserir no final da lista

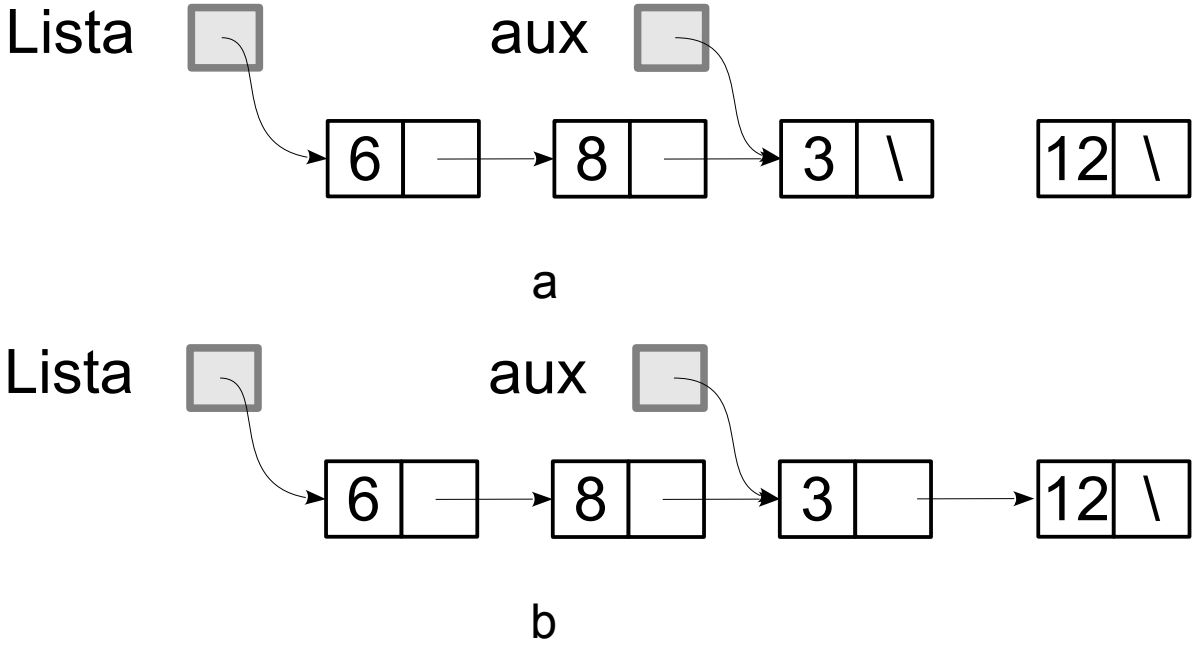


Inserir no final da lista



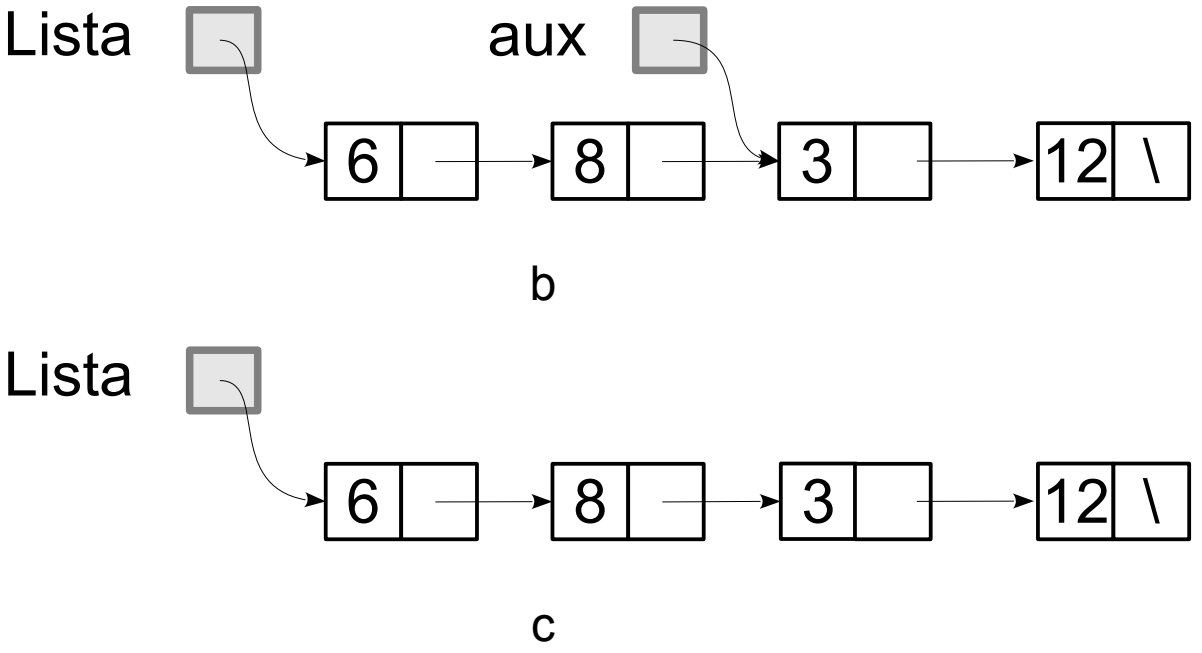
INSTITUTO FEDERAL
SÃO PAULO
Campus Guarulhos

Inserir no final da lista



INSTITUTO FEDERAL
SÃO PAULO
Campus Guarulhos

Inserir no final da lista



INSTITUTO FEDERAL
SÃO PAULO
Campus Guarulhos

Remover no início da lista

```
int remove_inicio(No** lista) {
   No* n;
   int v;

   if (*lista == NULL)
      return -1;

   n = *lista;
   *lista = n->prox;

   v = n->valor;
   free(n);
   return v;
}
```

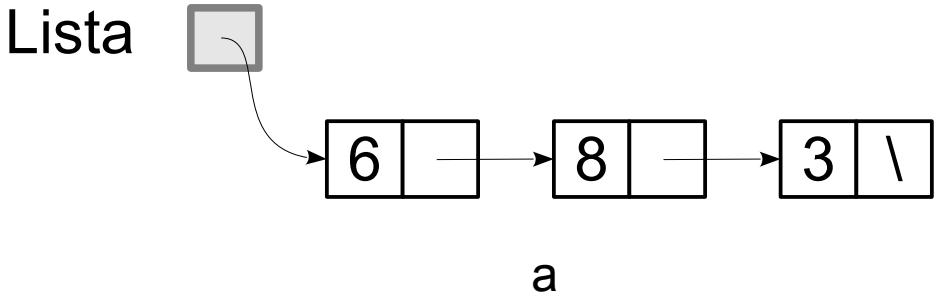
INSTITUTO FEDERAL
SÃO PAULO
Campus Guarulhos

Remover no início da lista

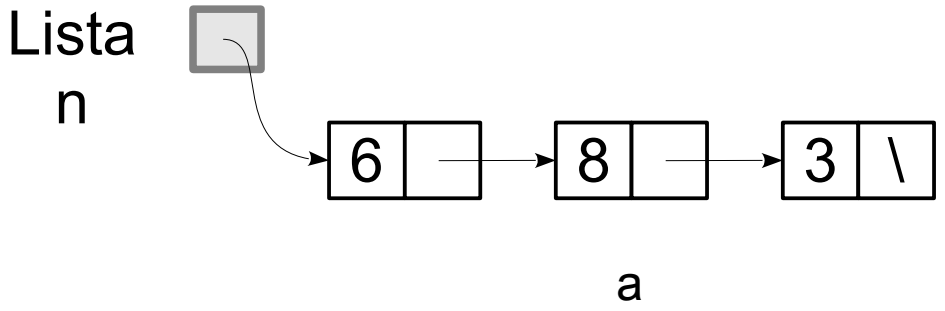
```
void main() {

   ...
   while (remove_inicio(&lista) != -1)
     imprime_lista(l);
}
```

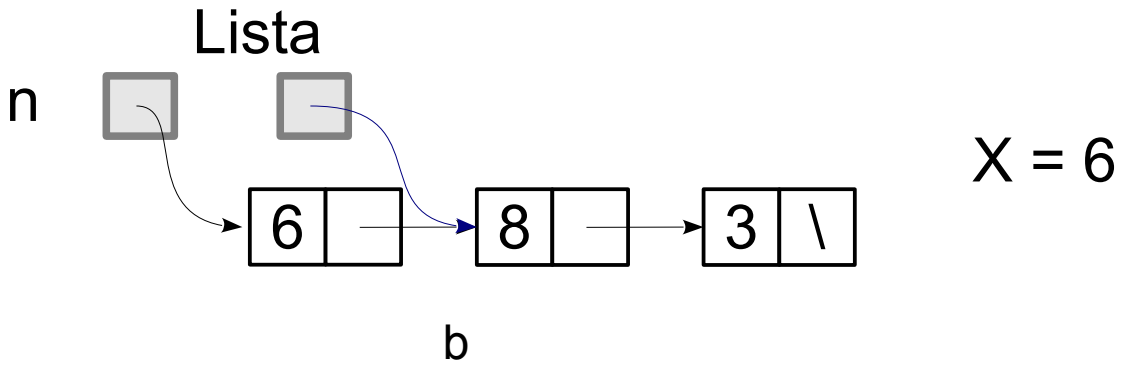
INSTITUTO FEDERAL
SÃO PAULO
Campus Guarulhos

Remover no início da lista

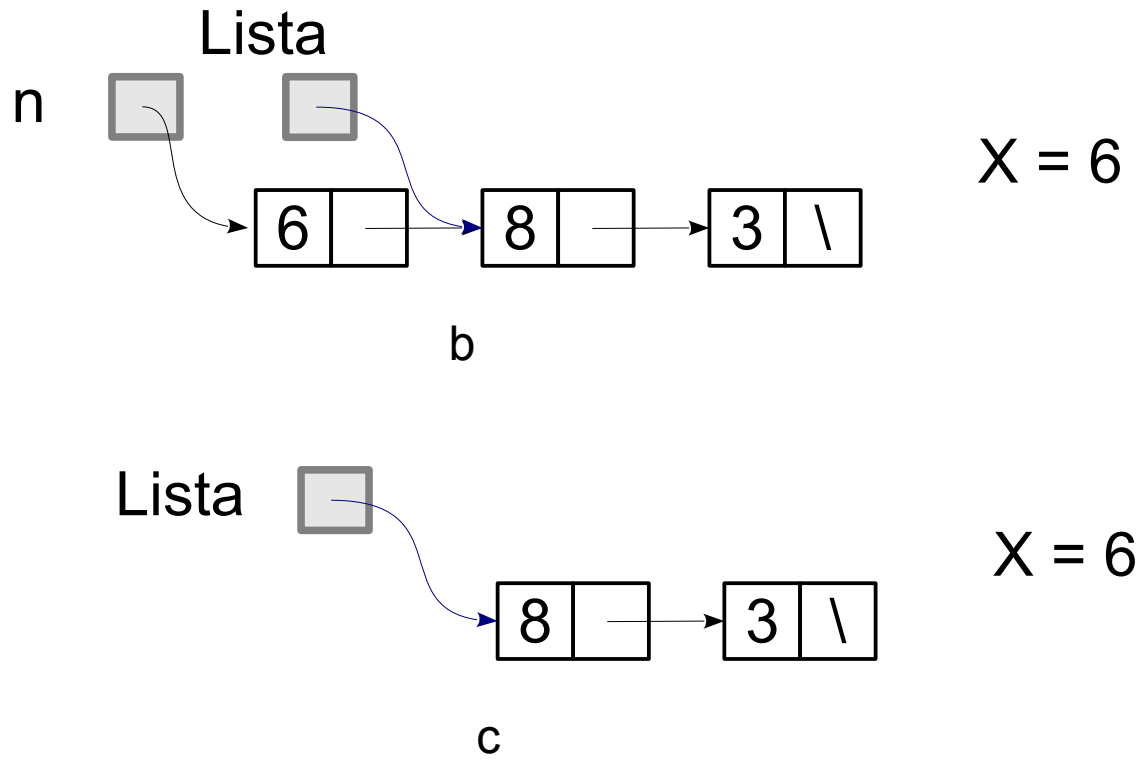


Remover no início da lista





Remover no início da lista



Remover no final da lista

```
int remove_final(No** lista) {
   No *n, *aux;
   int v;

   if (*lista == NULL)
      return -1;
   if ((*lista)->prox == NULL) {
      n = *lista;
      *lista = NULL;
   }
   //...
}
```

Remover no final da lista

```
int remove_final(No** lista) {
   // ...

   else {
      aux = *lista; /* Para no penúltimo nó */
      while (aux->prox->prox != NULL)
         aux = aux->prox;
      n = aux->prox;
      aux->prox = NULL;
   }
   v = n->valor;
   free(n);
   return v;
}
```

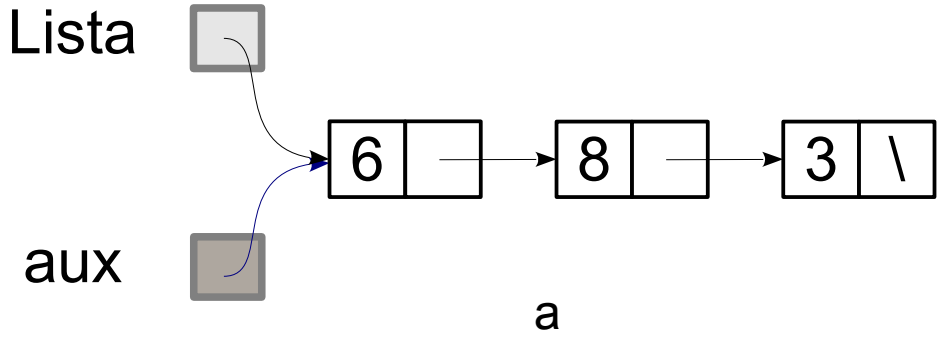
INSTITUTO FEDERAL
SÃO PAULO
Campus Guarulhos

Remover no final da lista

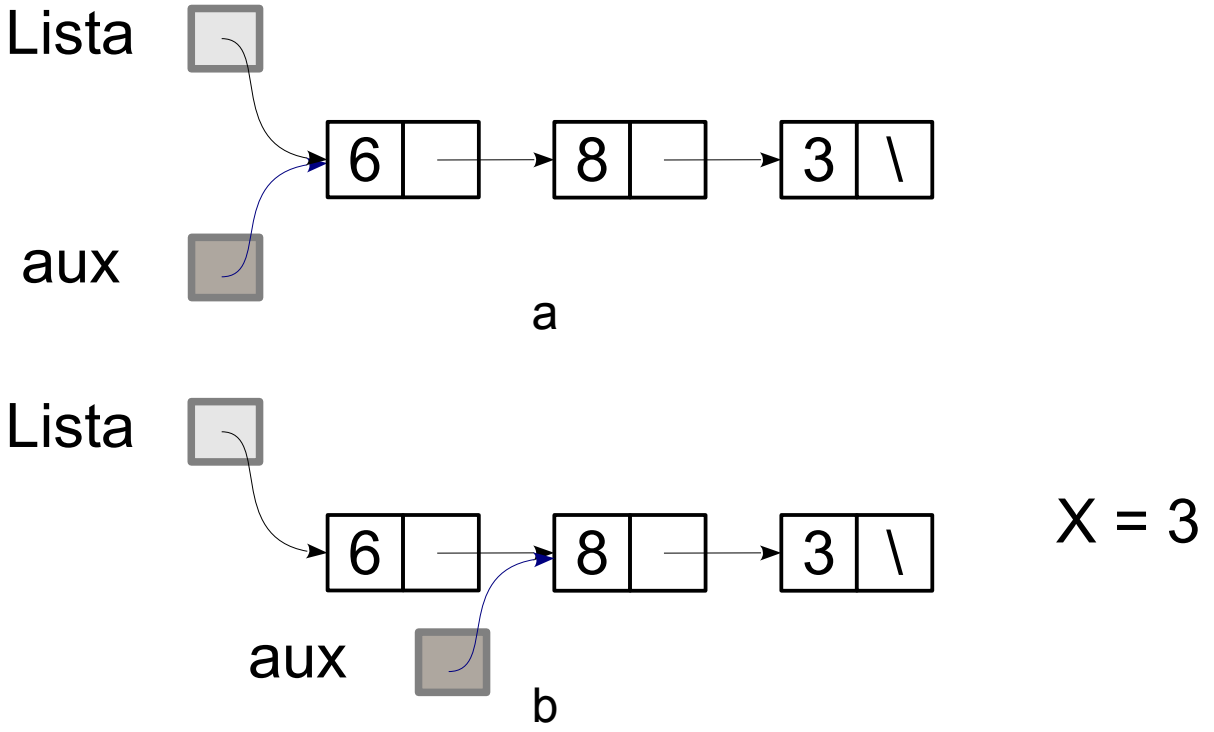
```
void main() {

   ...
   while (remove_final(&lista) != -1)
     imprime_lista(l);
}
```

INSTITUTO FEDERAL
SÃO PAULO
Campus Guarulhos

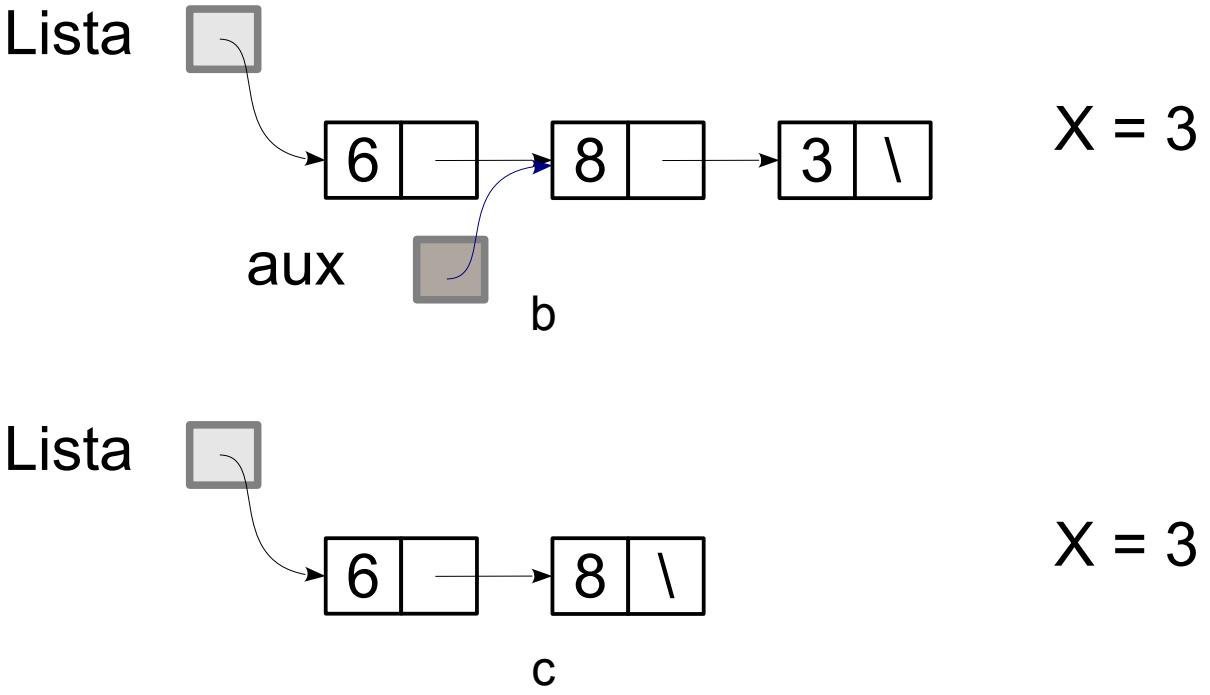
Remover no final da lista



Remover no final da lista



Remover no final da lista



Remover no final da lista