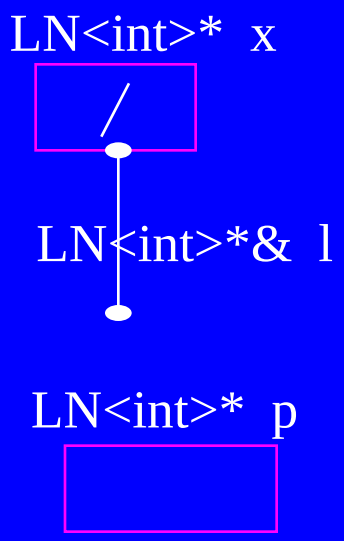
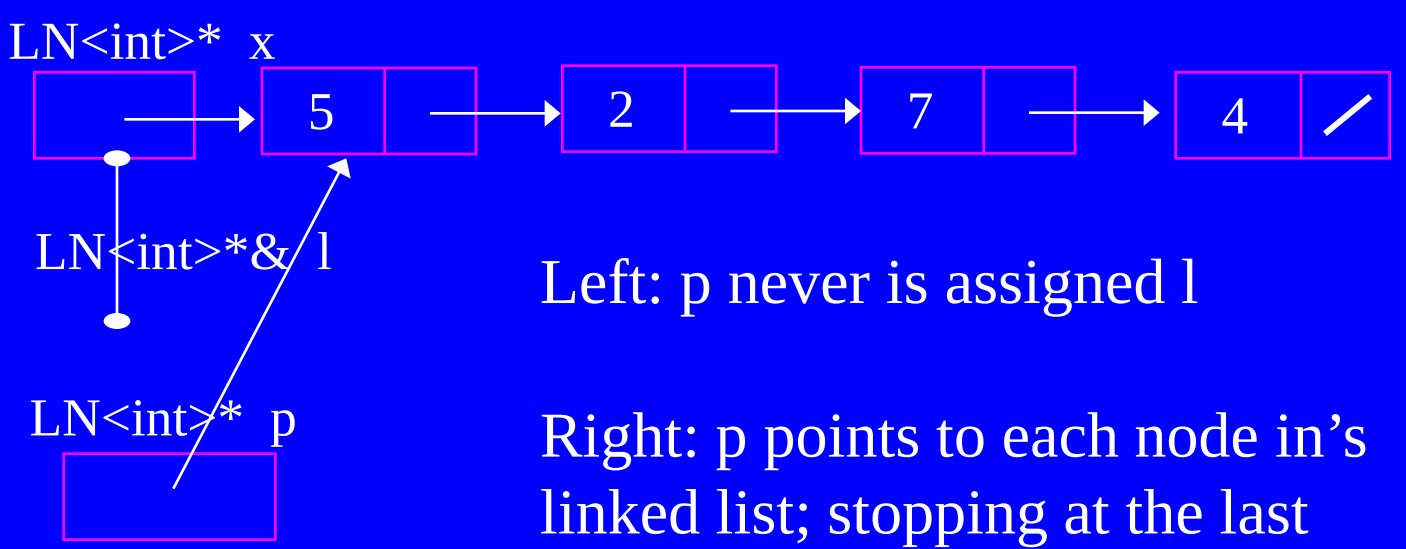
```
template<class T>
void add_rear(LN<T>*& l, T value) {
  if (l == nullptr)
    l = new LN<T>(value);
  else
    for (LN<T>* p = l; /*see body for termination*/; p = p->next)
      if (p->next == nullptr) {
        p->next = new LN<T>(value);
        break;
      }
}
```

Example: add_rear(x,10);



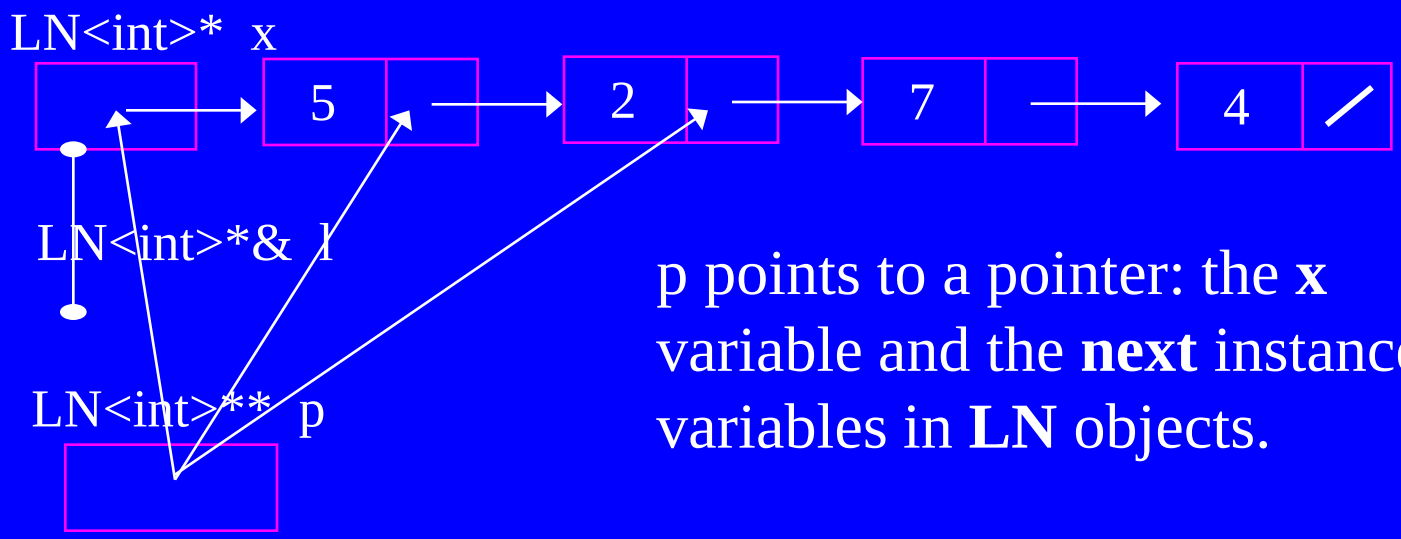


Left: p never is assigned l

Right: p points to each node in's linked list; stopping at the last

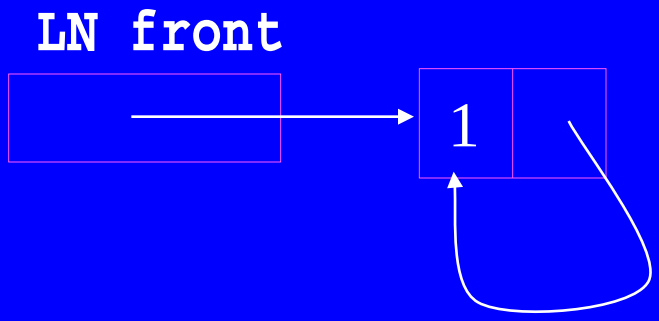
```
template<class T>
void remove_ptp (LN<T>*& l, T to_remove) {
  for (LN<T>** p =&l; (*p) != nullptr; p = &((*p)->next))
    if ((*p)->value == to_remove) {
      LN<T>* to_delete = *p;
      (*p) = (*p)->next;
      delete to_delete;
      break;
    }
}
```

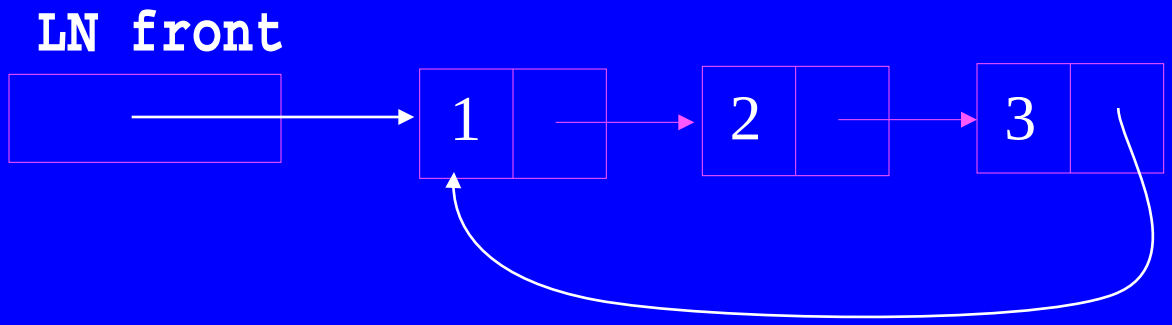
Example: remove_ptp(x,7);



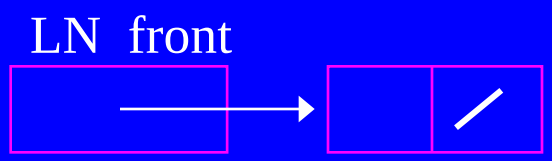
p points to a pointer: the **x** variable and the **next** instance variables in **LN** objects.

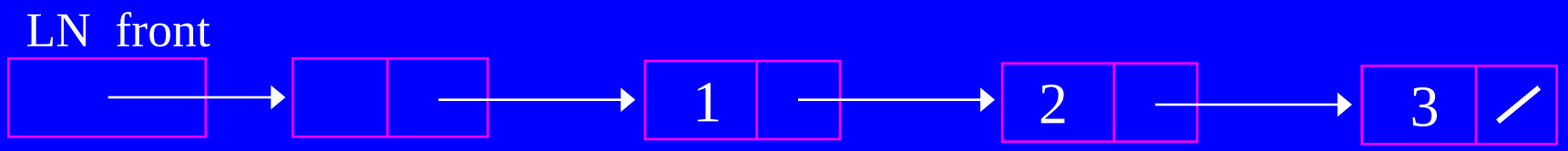
Circular Linked Lists



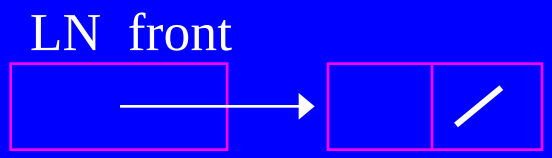


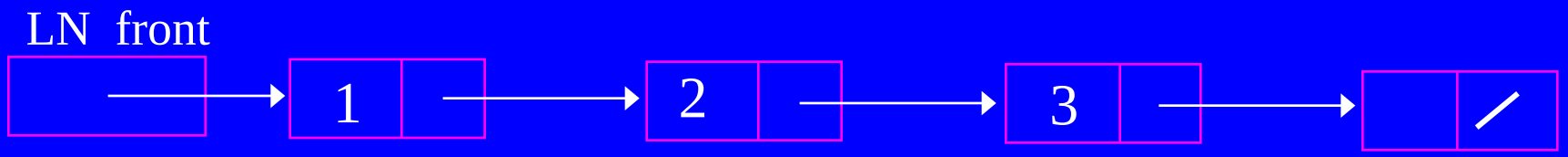
Header Linked Lists



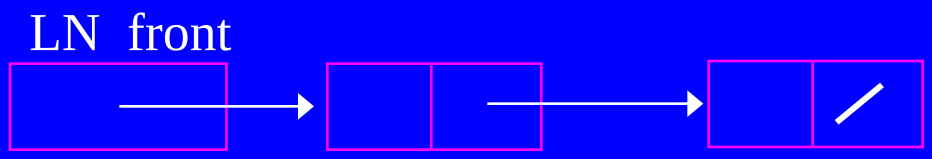


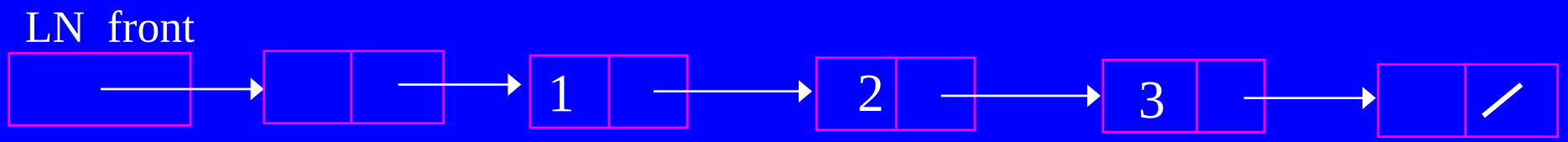
Trailer Linked Lists



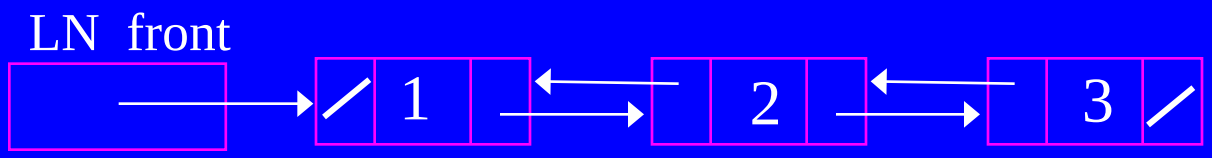


Header-Trailer Linked Lists





Doubly-Linked Lists



Doubly-Linked Header-Trailer Lists