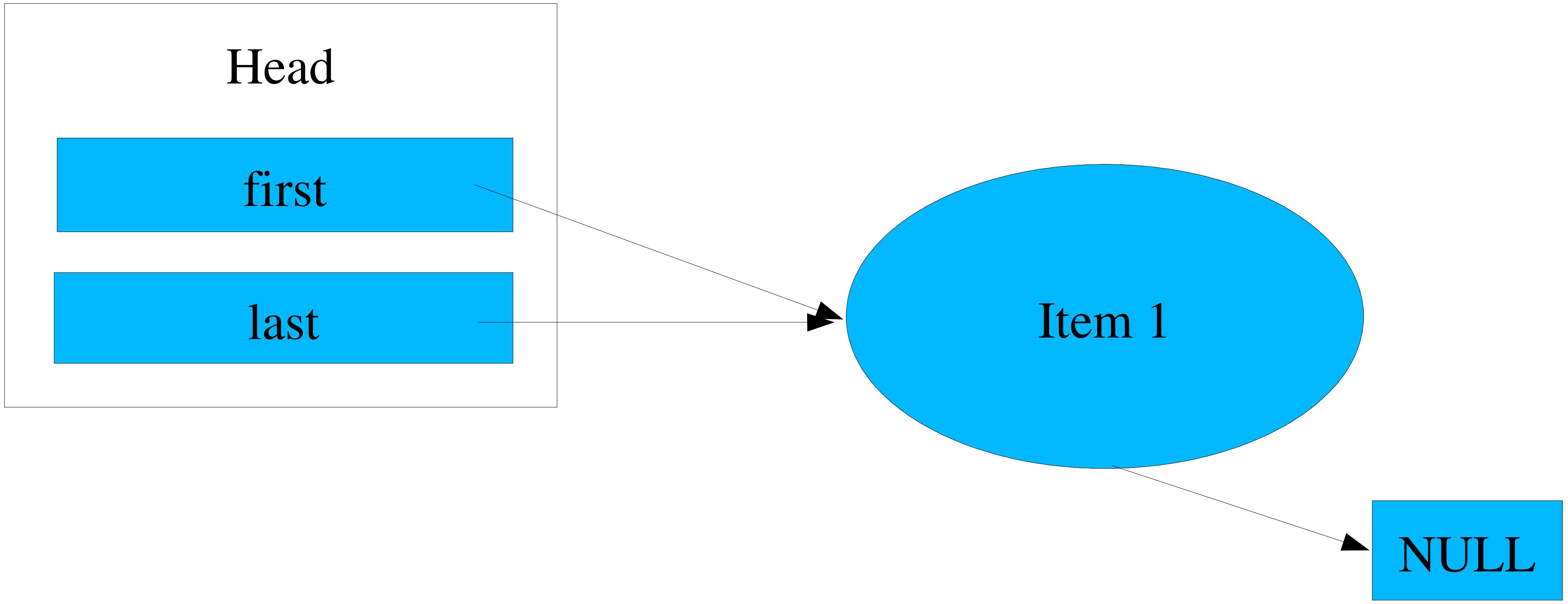
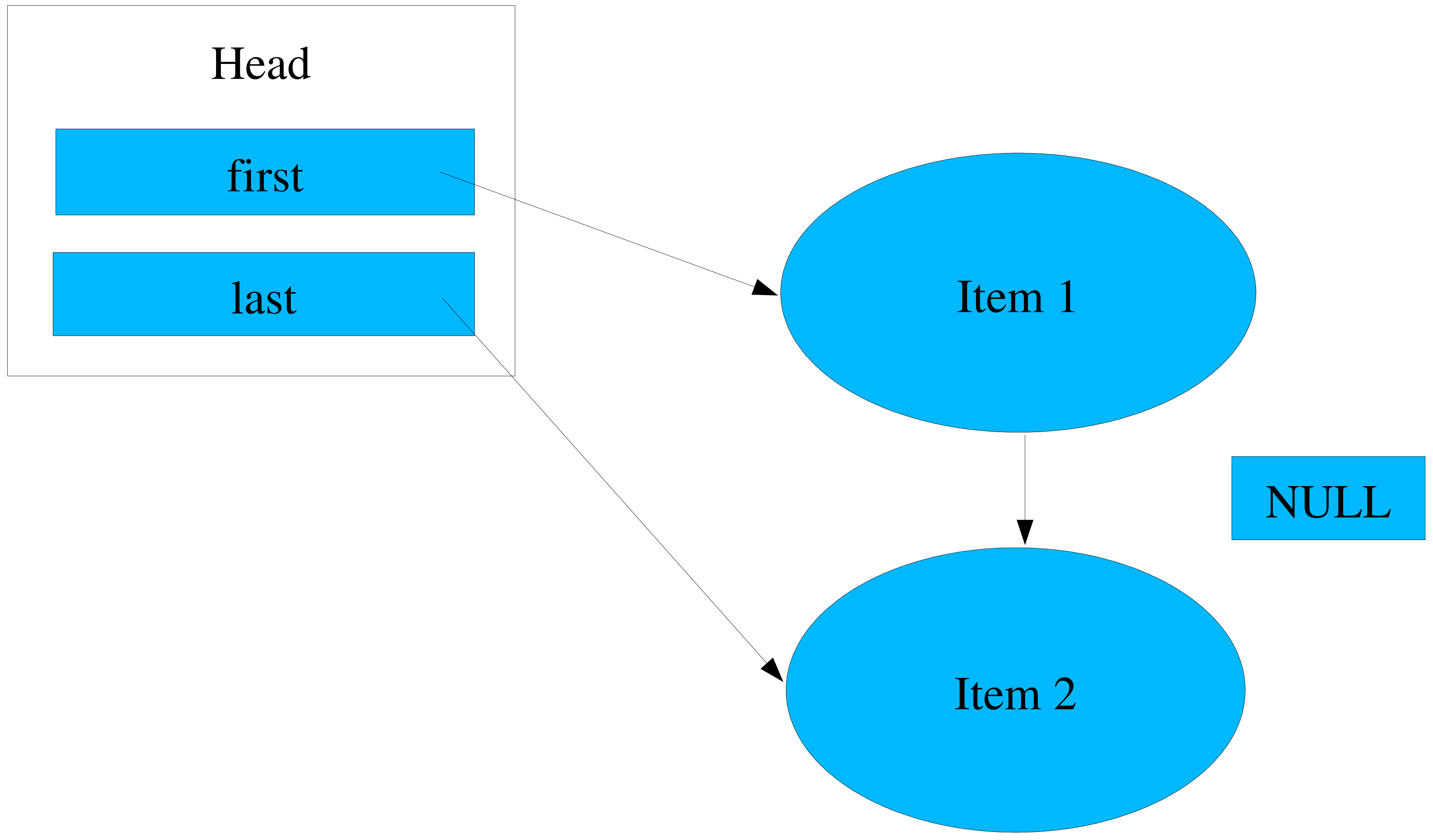
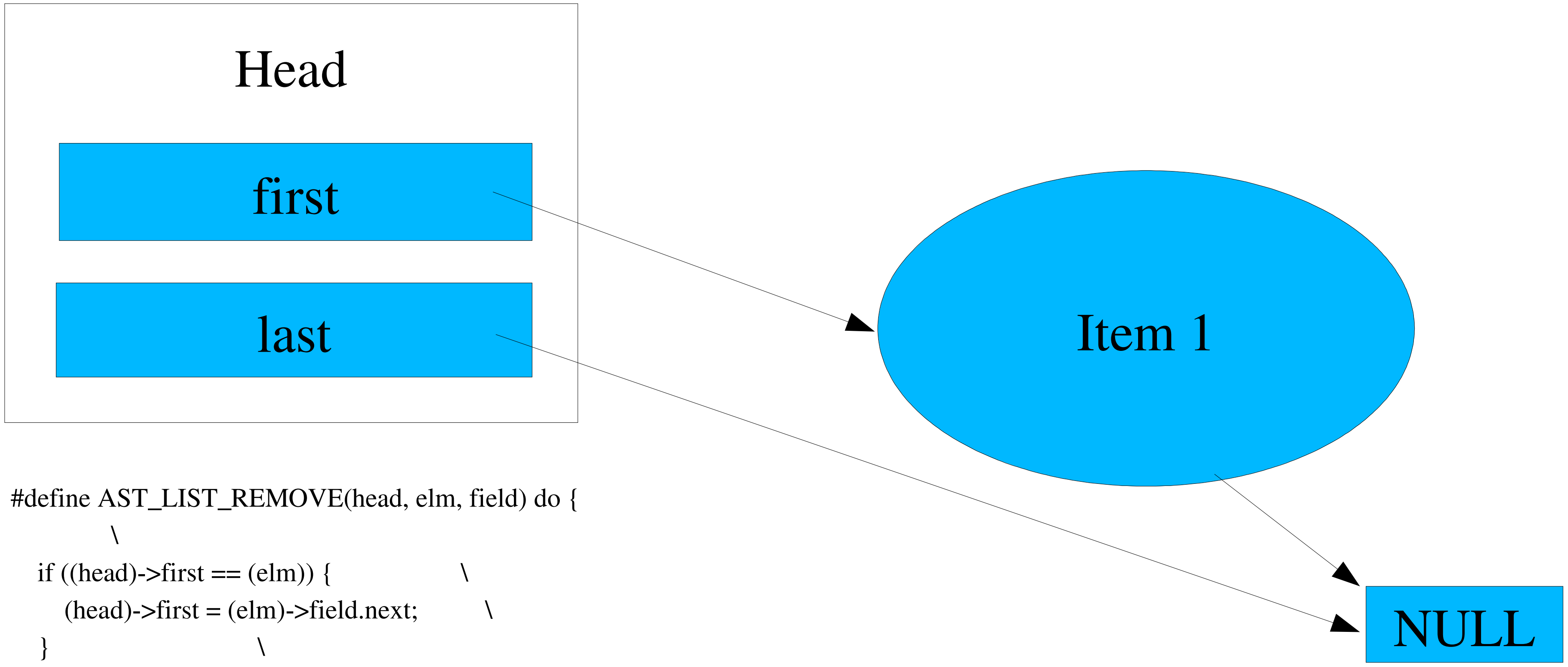

Head
first
last
Item 1
NULL

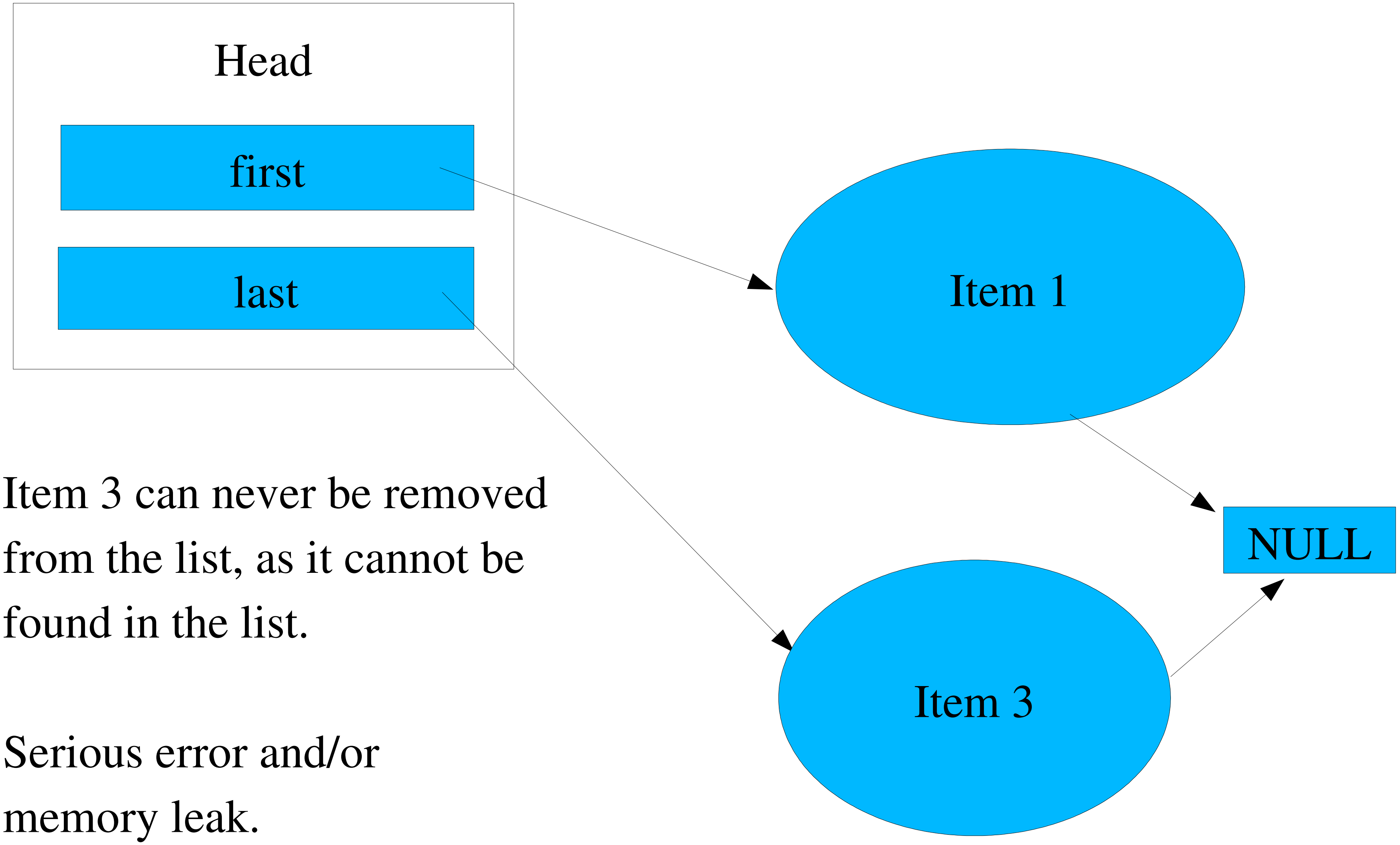


AST_LIST_INSERT_TAIL



```
#define AST_LIST_REMOVE(head, elm, field) do {
            \
    if ((head)->first == (elm)) {                  \
        (head)->first = (elm)->field.next;         \
    }                             \
    else {                             \
        typeof(elm) curelm = (head)->first;         \
        while (curelm->field.next != (elm))         \
            curelm = curelm->field.next;            \
        curelm->field.next = (elm)->field.next;         \
    }                             \
    if ((head)->last == elm)                  \
        (head)->last = NULL;                  \
} while (0)
```

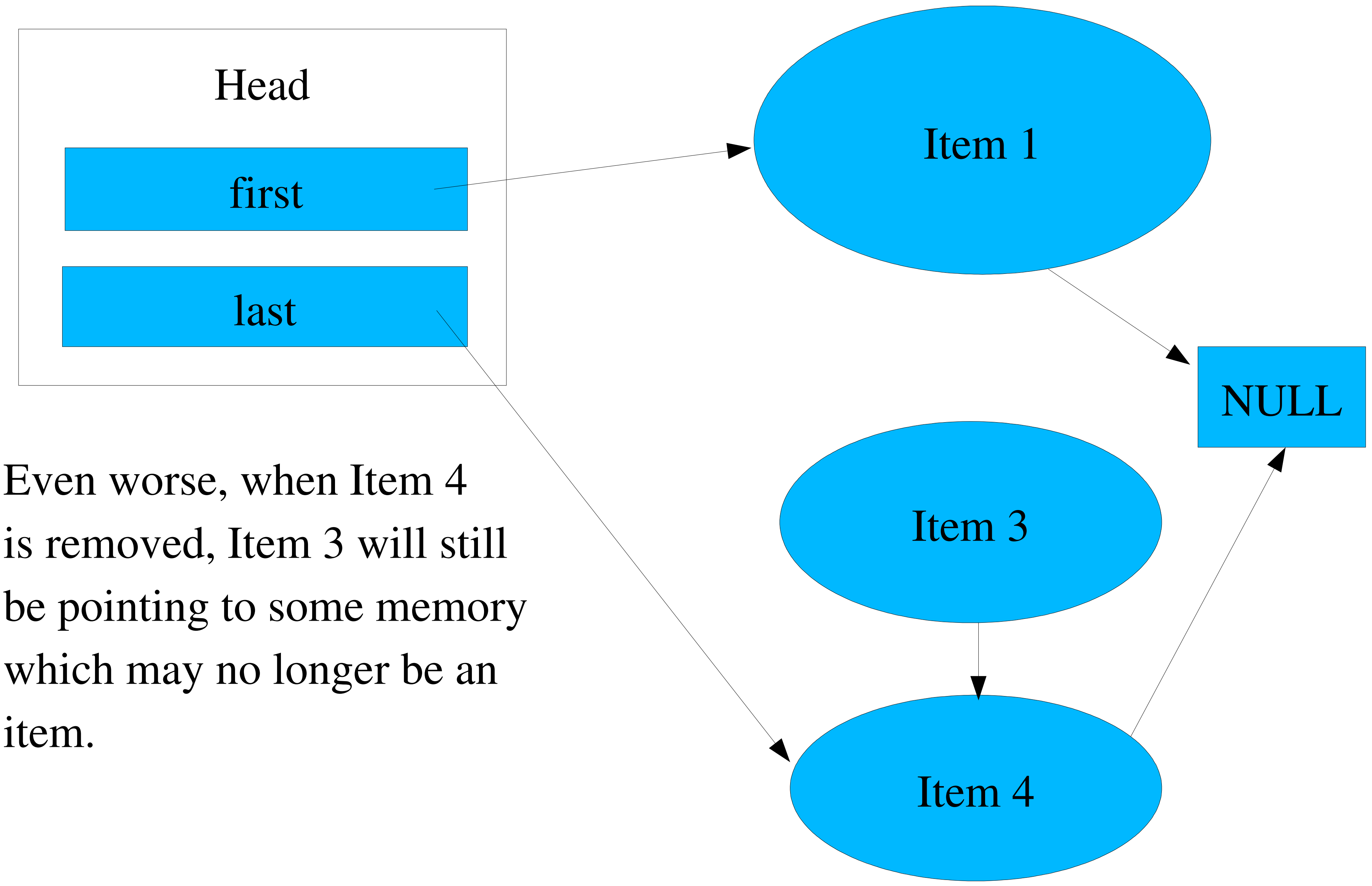
AST_LIST_REMOVE



Item 3 can never be removed from the list, as it cannot be found in the list.

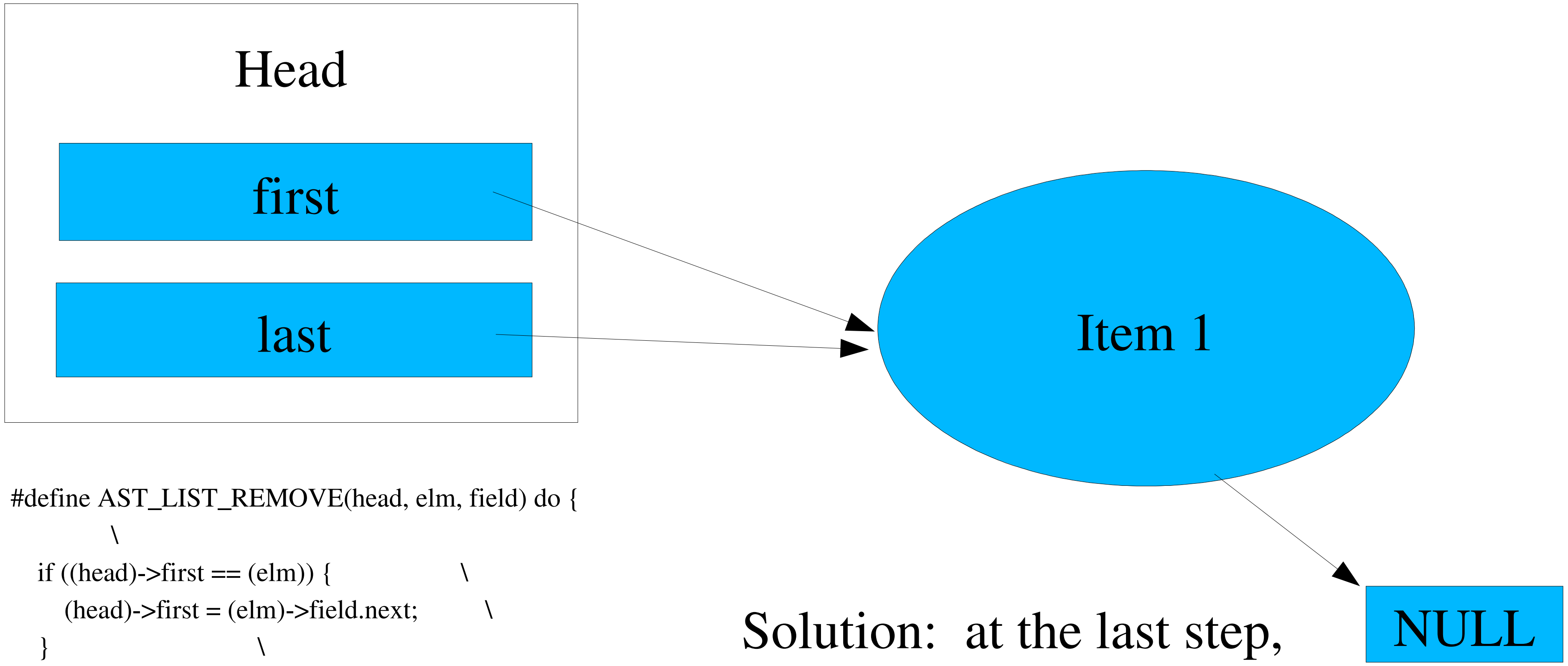
Serious error and/or memory leak.

AST_LIST_INSERT_TAIL



Even worse, when Item 4 is removed, Item 3 will still be pointing to some memory which may no longer be an item.

AST_LIST_INSERT_TAIL



```
#define AST_LIST_REMOVE(head, elm, field) do {
            \
    if ((head)->first == (elm)) {                  \
        (head)->first = (elm)->field.next;         \
    }                              \
    else {                             \
        typeof(elm) curelm = (head)->first;        \
        while (curelm->field.next != (elm))        \
            curelm = curelm->field.next;           \
        curelm->field.next = (elm)->field.next;        \
    }                              \
    if ((head)->last == elm)                 \
        (head)->last = NULL;                 \
} while (0)
```

Solution: at the last step, we must iterate through the list to find the last item and point last to it, rather than setting last to NULL.

## AST_LIST_REMOVE