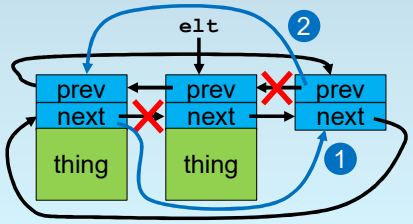
Two Steps for Removal from List

```
elt->prev->next = elt->next;//step 1
elt->next->prev = elt->prev;//step 2
```



Can Also Provide Other Access Methods

We can also provide other access methods.

For example, a routine to
find the first element in a list.

```
// Returns a pointer to the first
// element in a list, or NULL for
// an empty list.
void* dl_first
    (double_list_t* head);
```


One Line Needed to Find the First Element in a List

Such a routine is not hard to write:

```
// Returns a pointer to the first
// element in a list, or NULL for
// an empty list.
void* dl_first
    (double_list_t* head)
{
    return (head == head->next ?
            NULL : head->next);
}
```

Is the list empty?

Can We Provide a Way to Iterate Over "Things"?

What about iterating over the list elements?

Let's say that we want
- to **execute some code**
- **for every "thing"** in a list.

Again, we want to
- **avoid exposing** the **details** of our list implementation, and
- **avoid requiring** many **copies of iteration control** code.

