Passing Linked Lists

CSCI-2270

Elizabeth Boese

Linked Lists

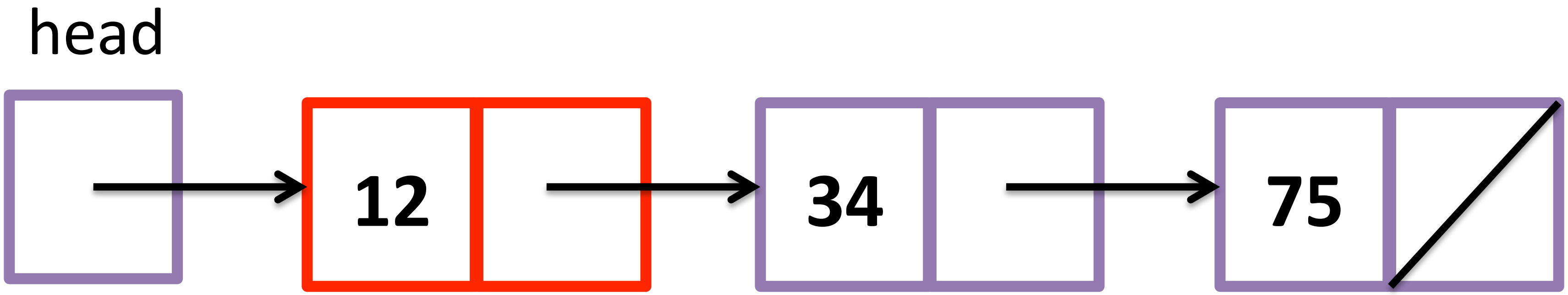
- What happens when you call a function to delete a node in the list,

  and that node happens to be first in the list?

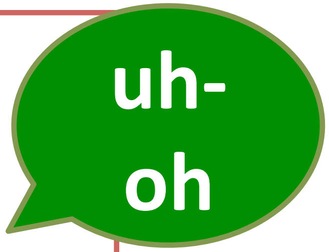
Linked Lists

What happens when you call a function to delete a node in the list,

and that node happens to be first in the list?



```
void deleteFirstNode(Node* ptr)
{
    Node *delNode = ptr;
    ptr = delNode->next;
    delete delNode;
}
```



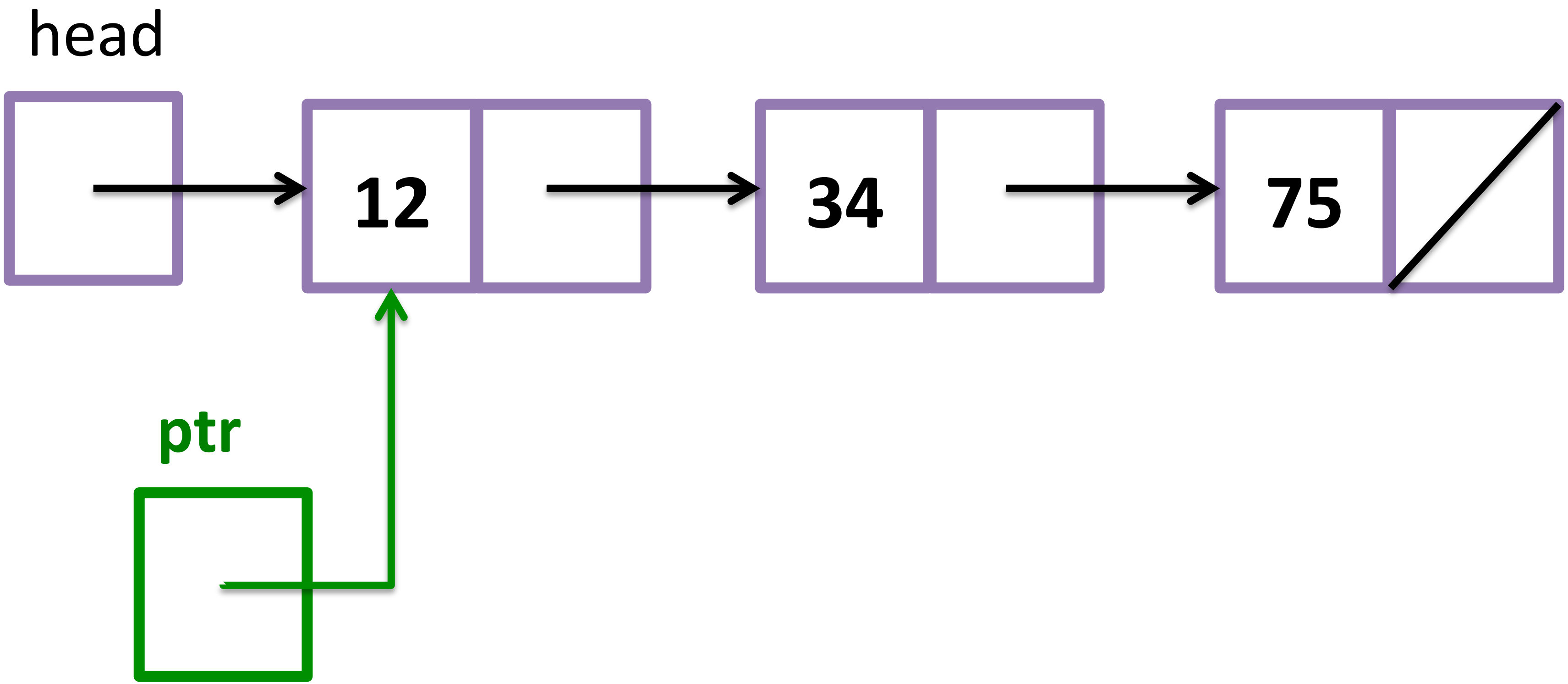
Linked Lists

What happens when you call a function to delete a node in the list,

and that node happens to be first in the list?

uh-oh

```
void deleteFirstNode(Node* ptr)
```



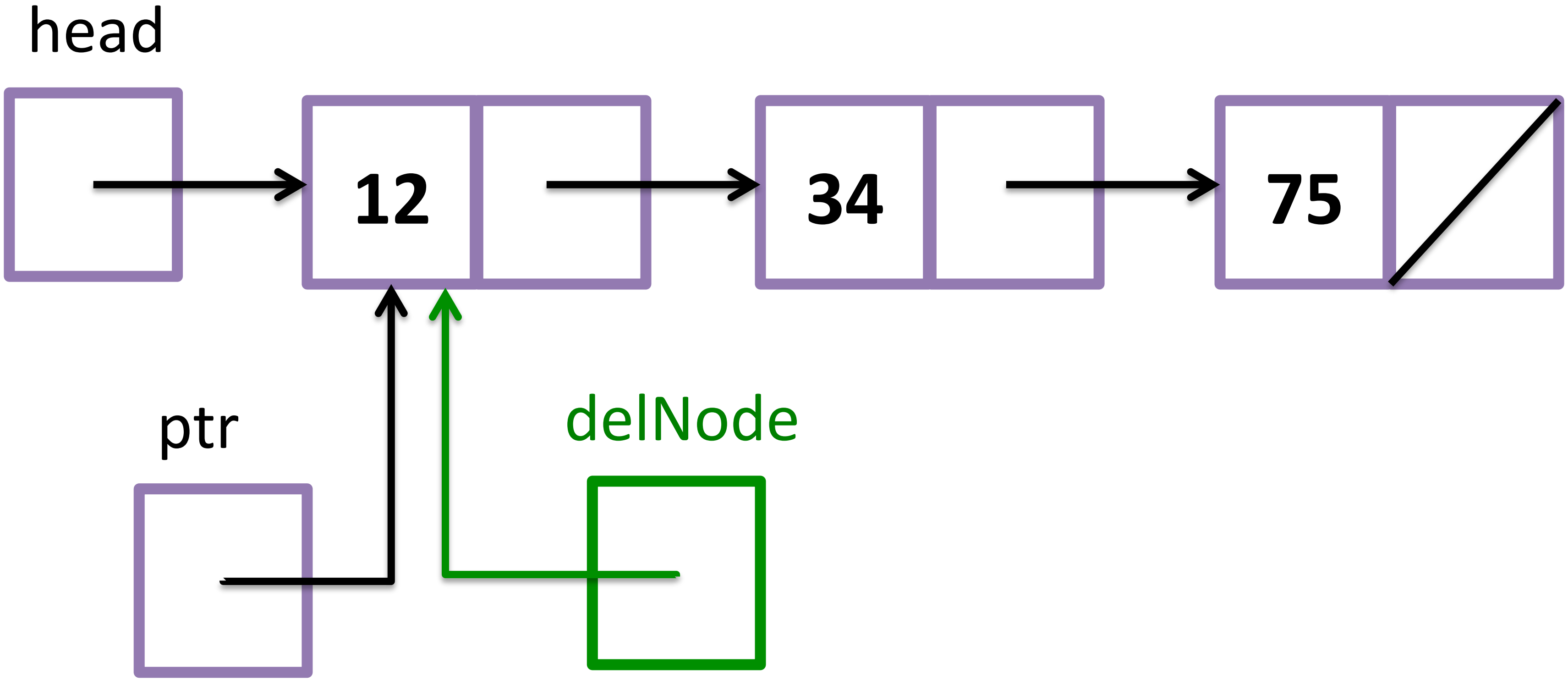
Linked Lists

- What happens when you call a function to delete a node in the list,

  and that node happens to be first in the list?

uh-oh

```
Node *delNode = ptr;
```

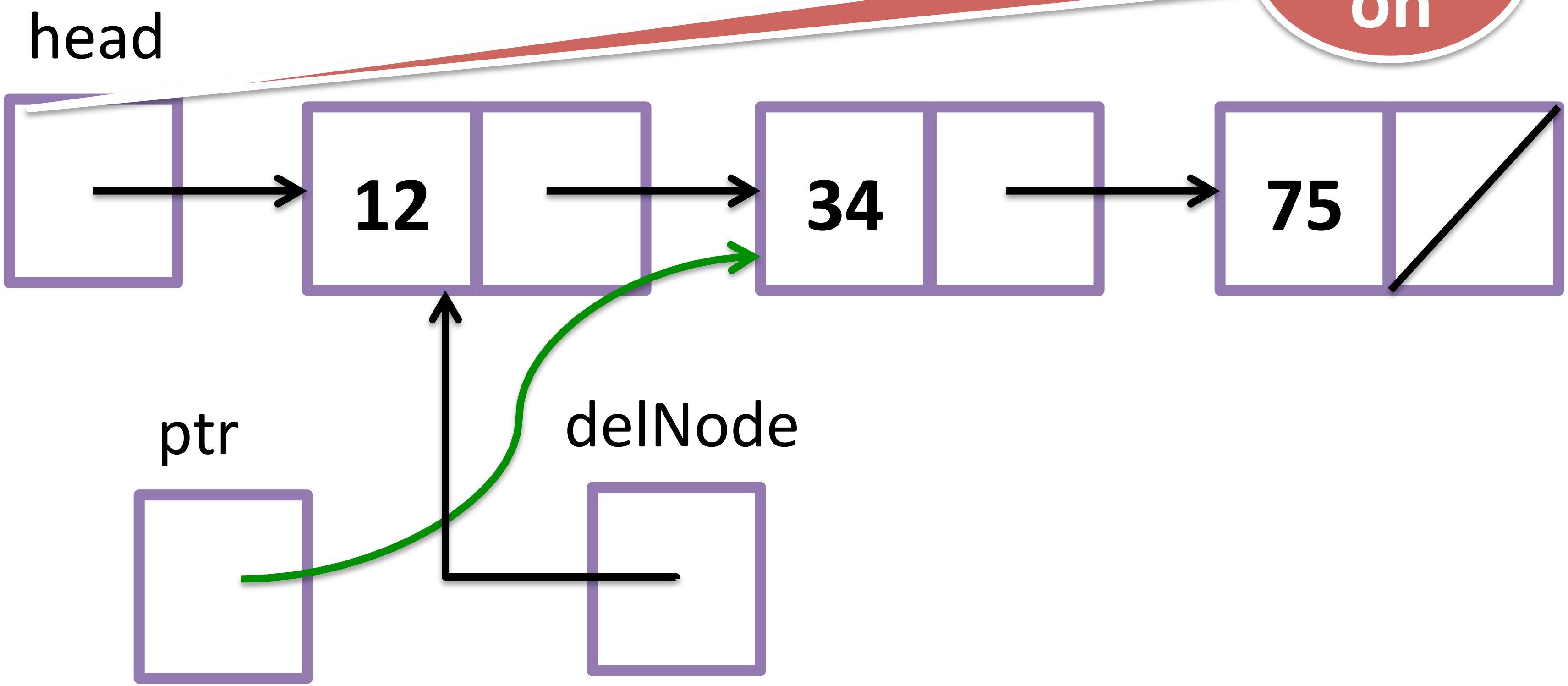


Linked Lists

- What happens when you call a function to delete a node in the list,
  and that node happens to be first in the list?

```
ptr = delNode->next;
```

uh-oh



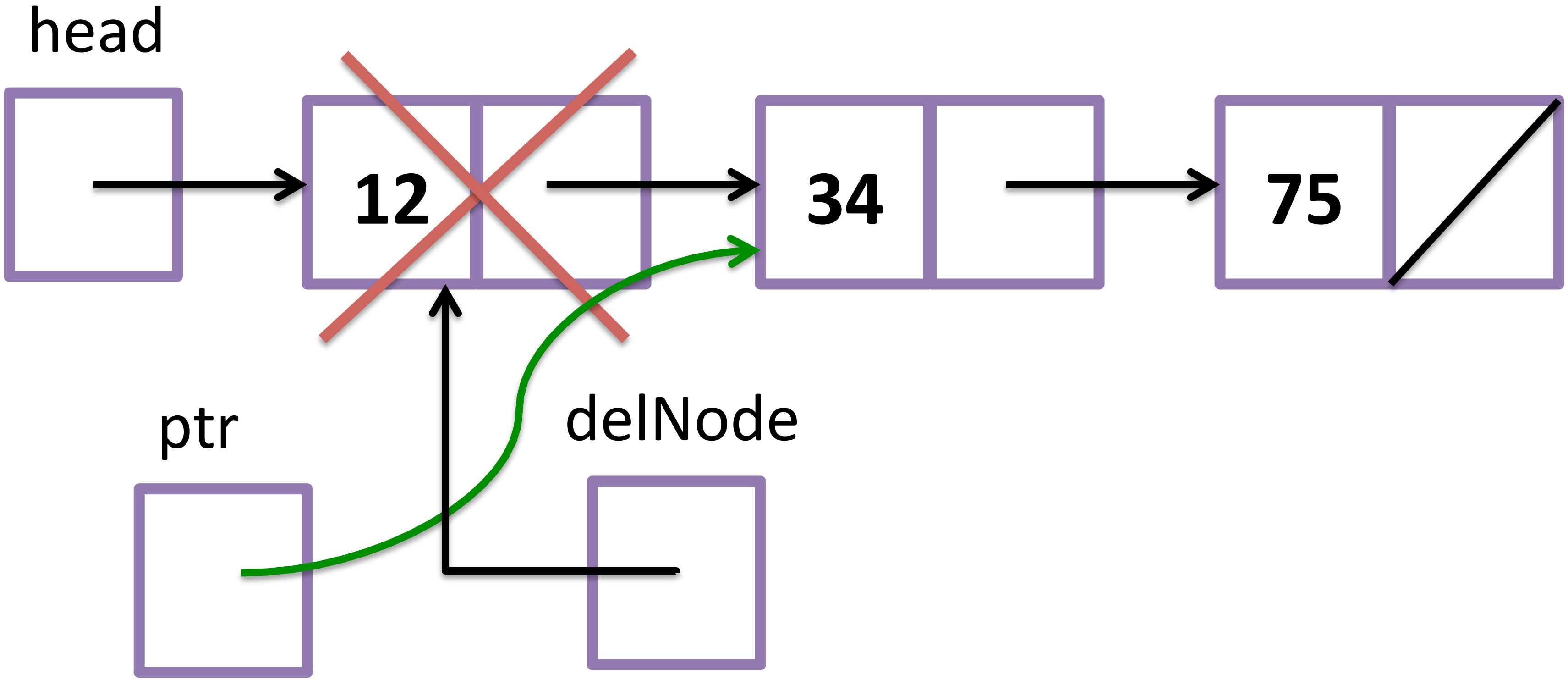
Linked Lists

- What happens when you call a function to delete a node in the list,

  and that node happens to be first in the list?

uh-oh

```
delete delNode;
```



Passing Linked Lists

- Functions that operate on a linked list either
  - do not modify the head pointer
  - do modify the head pointer

Passing Linked Lists

- If function does not modify the head pointer
  - can just pass the head node pointer
    - e.g., traverse the list (print, find, edit values in nodes...)

Passing Linked Lists

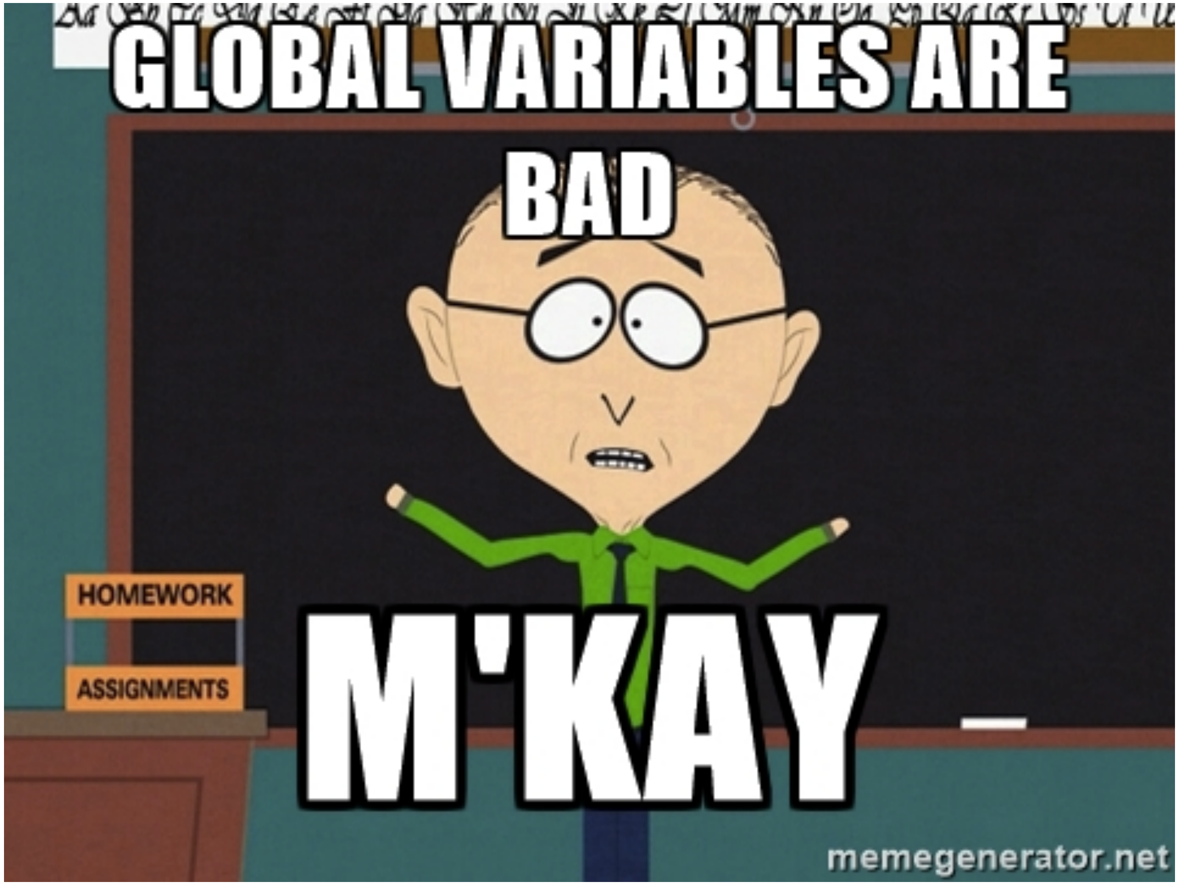
If a function does need to modify the head node

- – Insert at beginning of list
- – Delete first node in list
- – etc.

- How?

Functions that modify Linked Lists

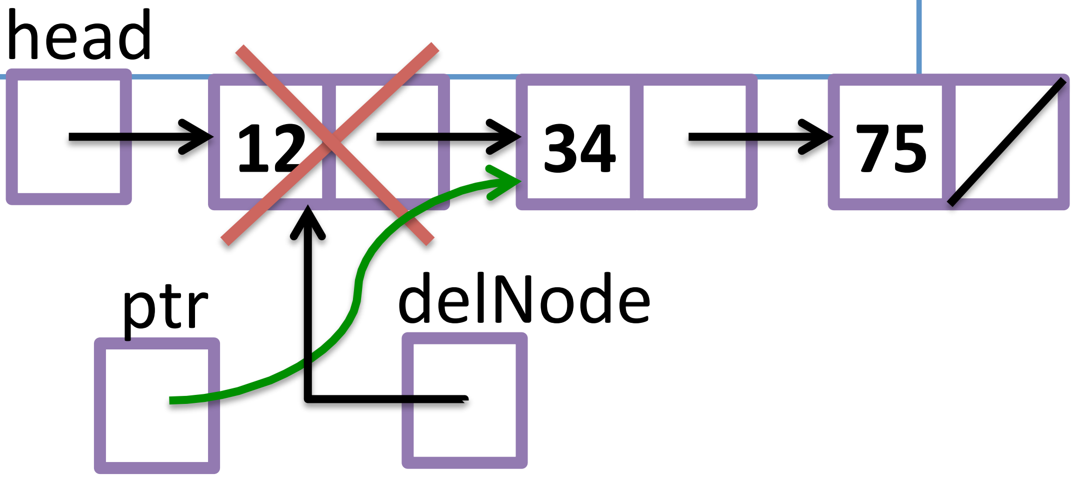
Option 1: Global head variable



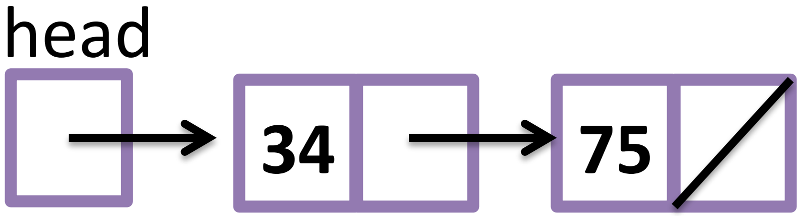
Functions that modify Linked Lists

Option 2: Return the head pointer

```
Node* deleteFirstNode(Node* ptr)
{
    ...
    return ptr;
}
```



```
head = deleteFirstNode(head);
```



Functions that modify Linked Lists

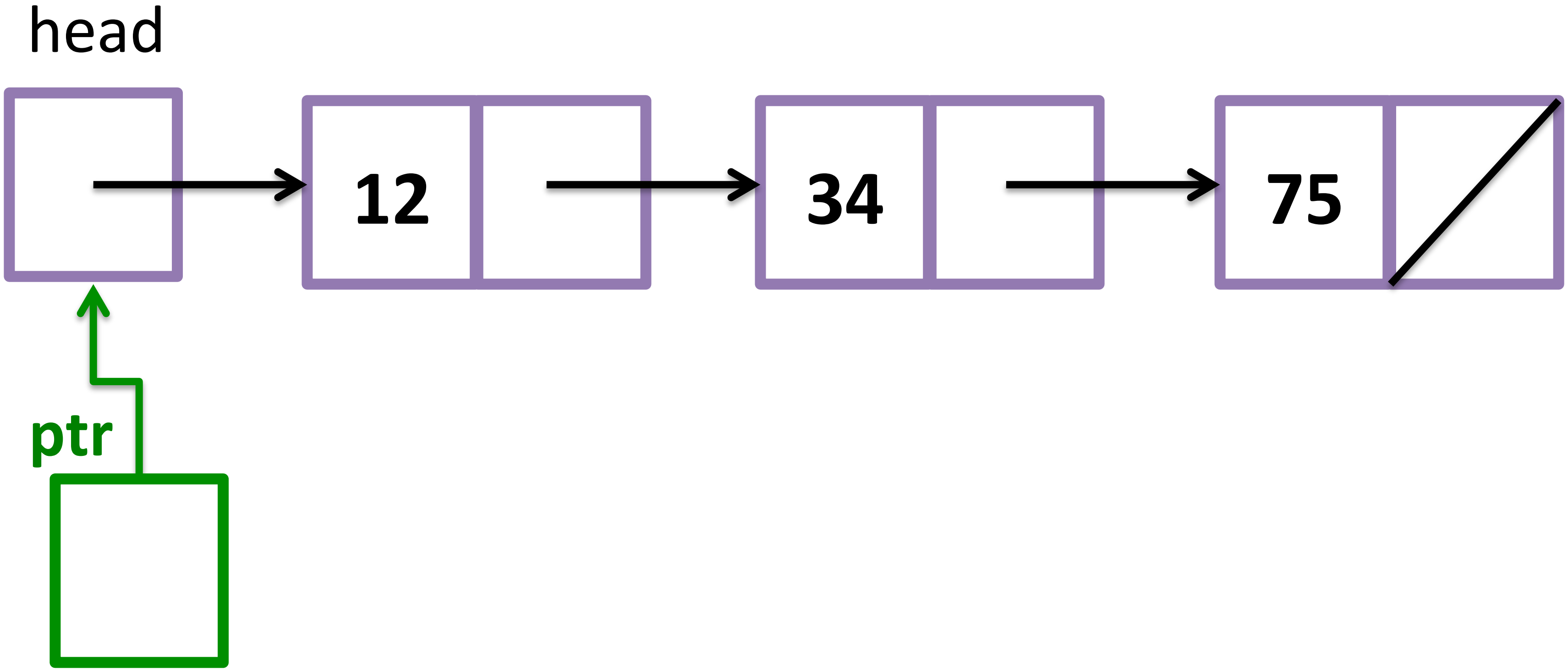
Option 3: Double-pointer

```
void deleteFirstNode(Node **ptr)
```

Linked Lists

Double-pointer

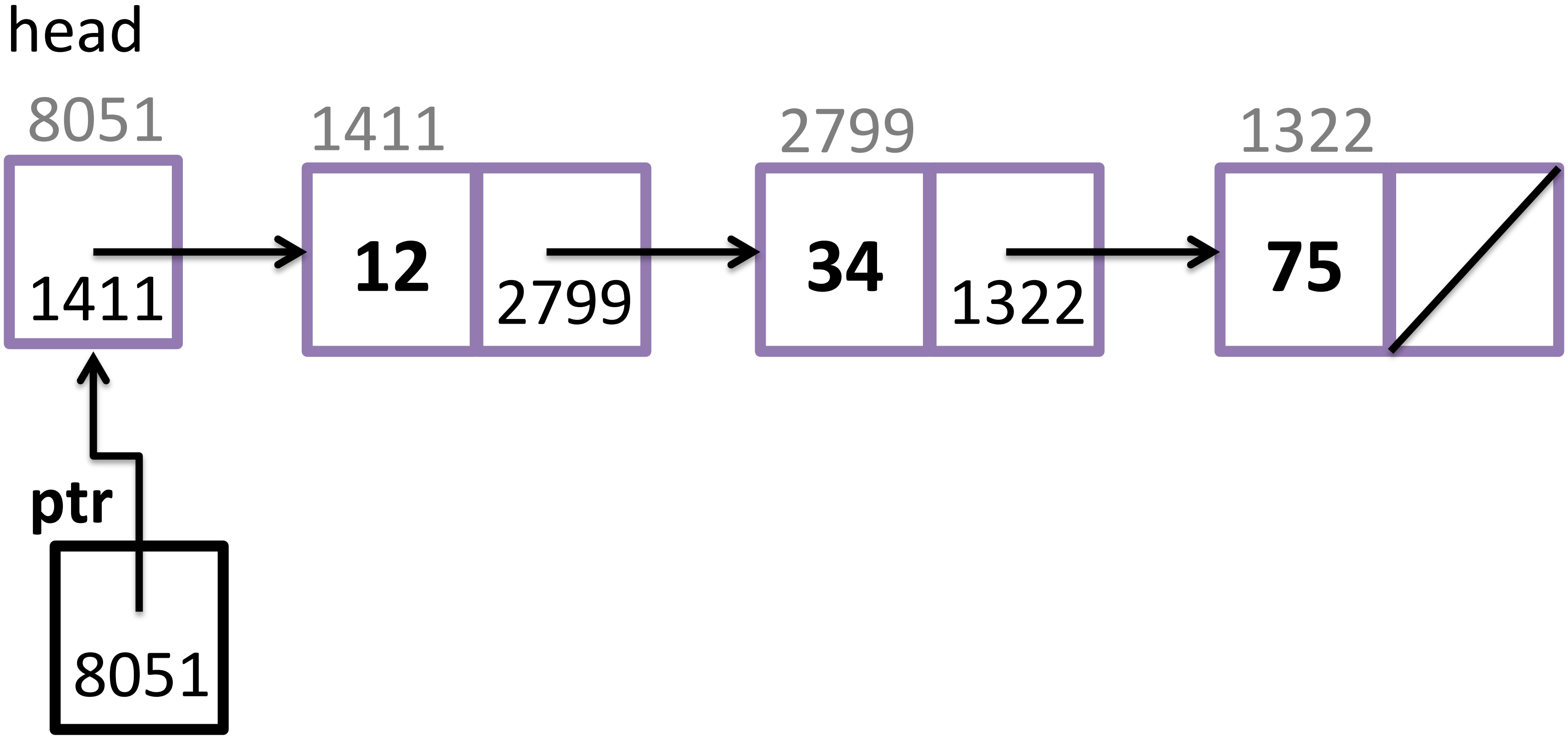
```
void deleteFirstNode(Node** ptr)
```



Linked Lists

Double-pointer

```
void deleteFirstNode(Node** ptr)
```

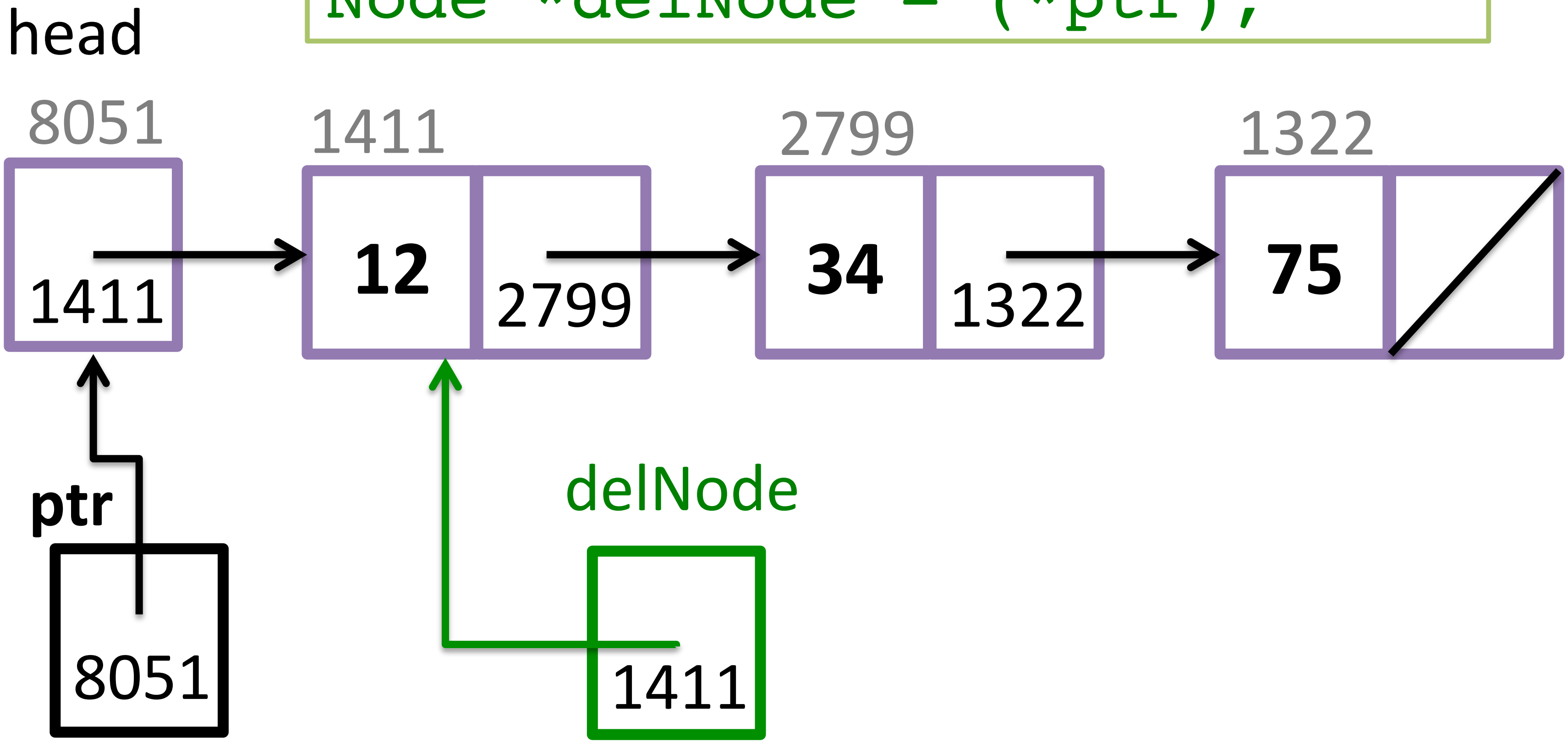


Linked Lists

Double-pointer

```
void deleteFirstNode(Node** ptr)
```

```
Node *delNode = (*ptr);
```

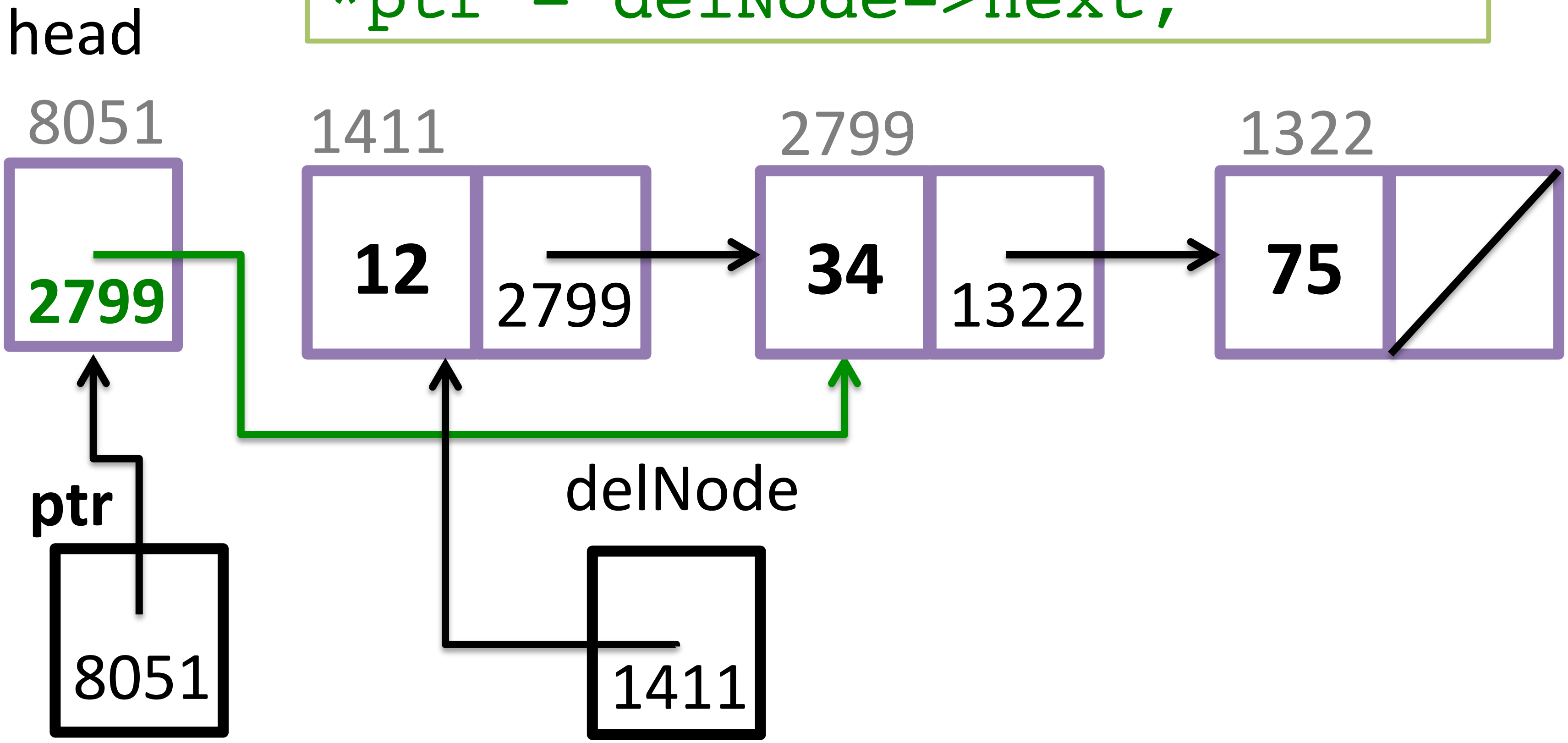


Linked Lists

Double-pointer

```
void deleteFirstNode(Node** ptr)
```

```
*ptr = delNode->next;
```

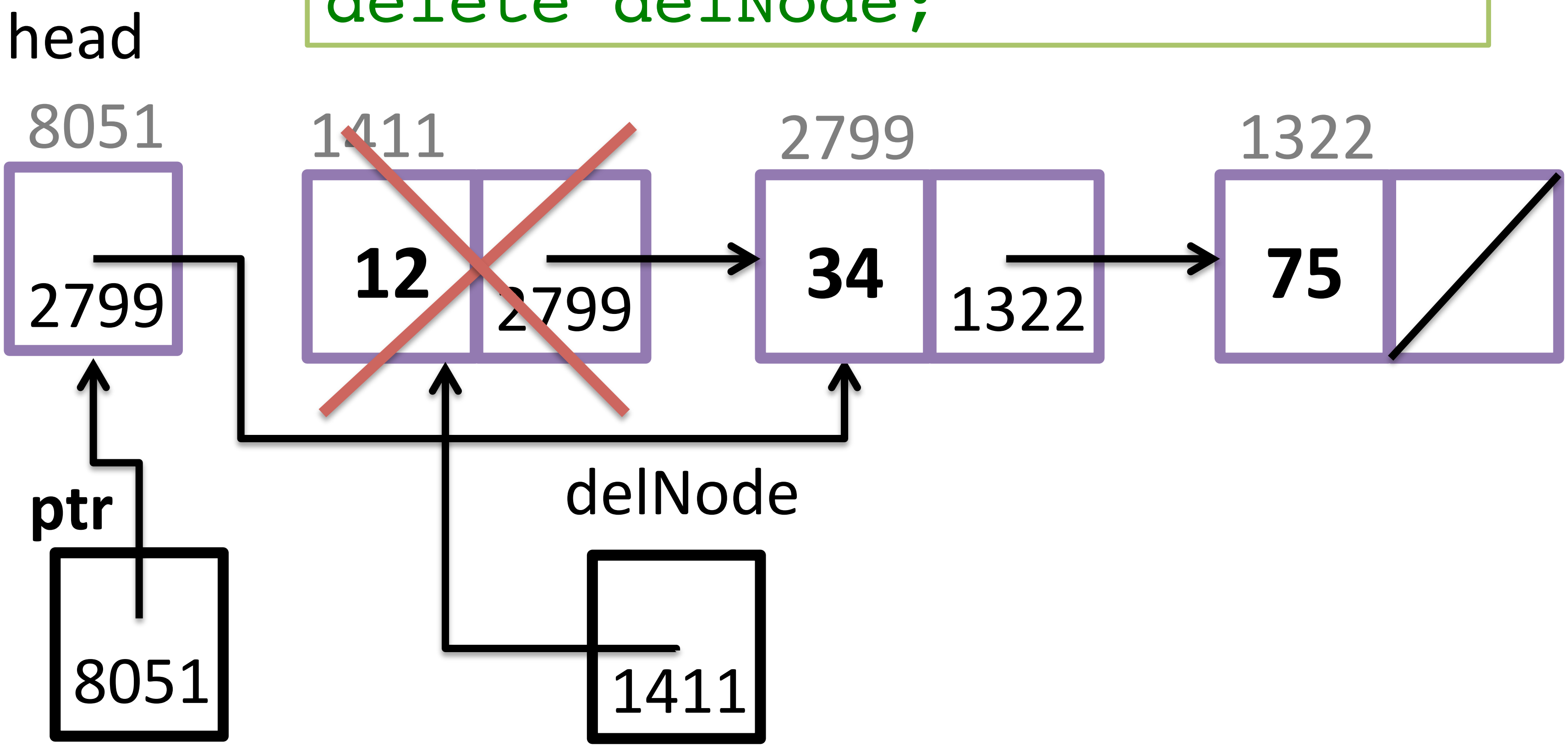


Linked Lists

Double-pointer

```
void deleteFirstNode(Node** ptr)
```

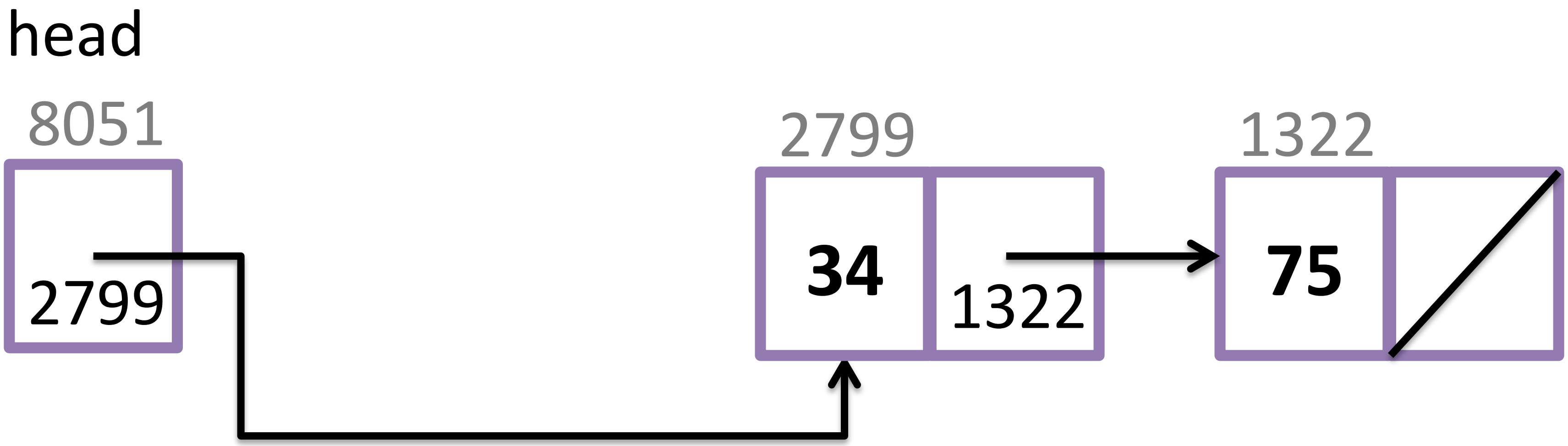
```
delete delNode;
```



Linked Lists

Double-pointer – After call to function

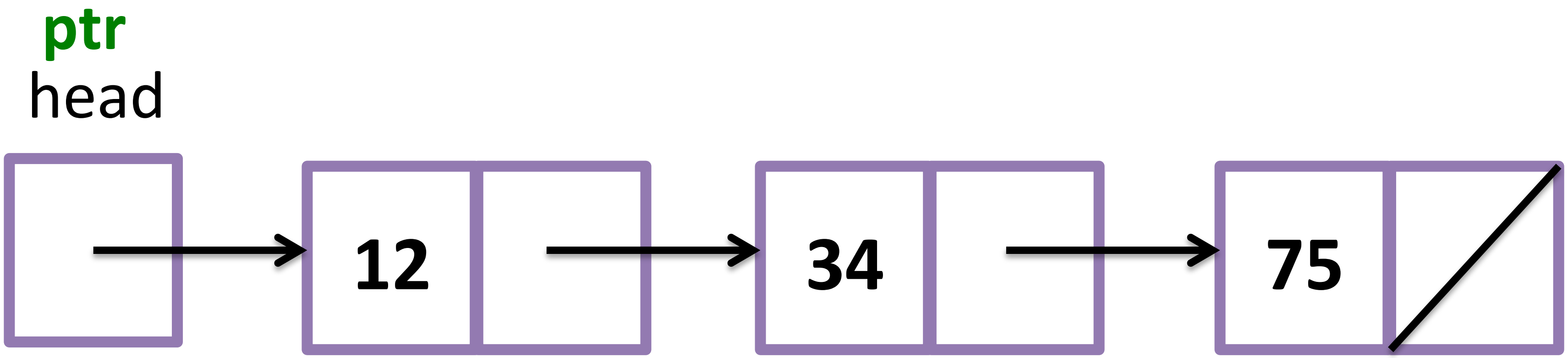
```
void deleteFirstNode(Node** ptr)
```



Functions that modify Linked Lists

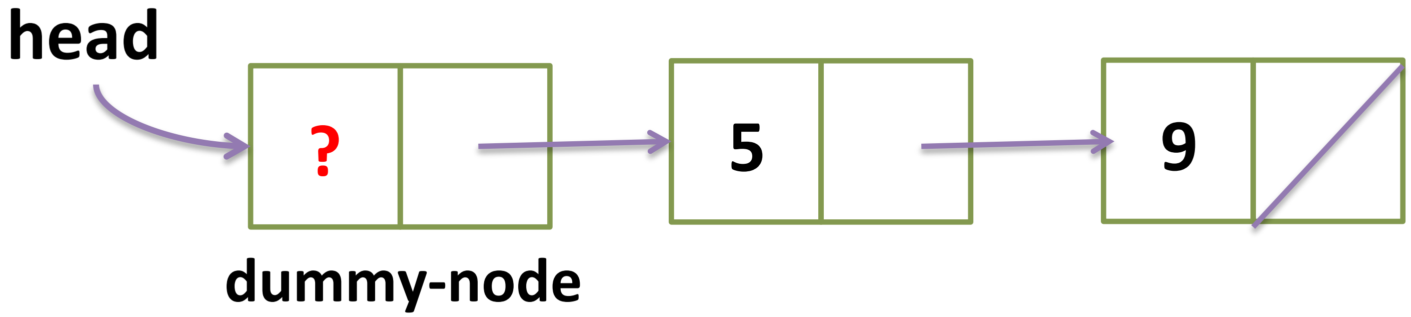
Option 4: Address of head Pointer

```
void deleteFirstNode(Node*& ptr)
```



Functions that modify Linked Lists

Option 5: Use a dummy-head node



can be really confusing code

Questions to Ponder

1. Write a function to:
   a) add a node to the beginning of a list
   b) delete a node from the beginning of a list