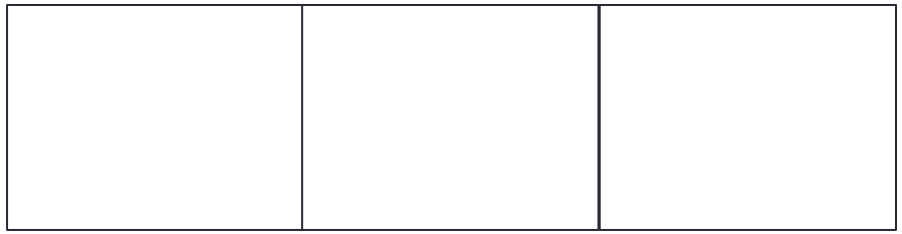
INTRO TO LINKED LISTS

Problem Solving with Computers-II

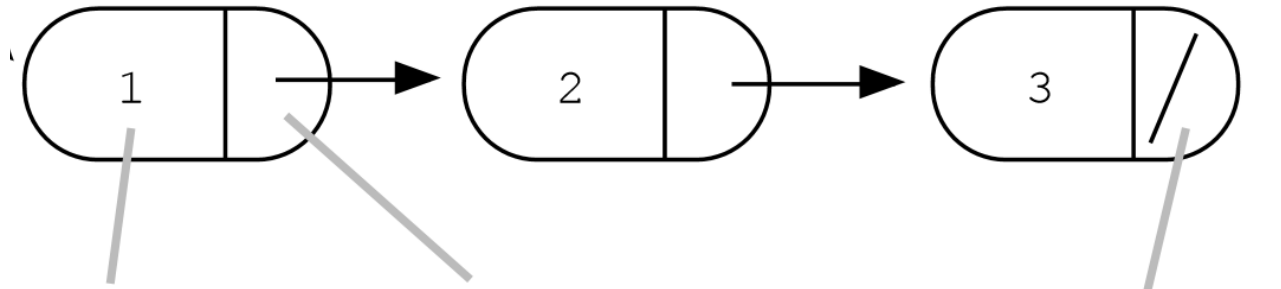
Linked list vs Array

Array



Defining the type Node

The overall list is built by connecting the nodes together by their next pointers. The nodes are all allocated in the heap.



Each node stores one data element (int in this example).

Each node stores one next pointer.

The next field of the last node is NULL.

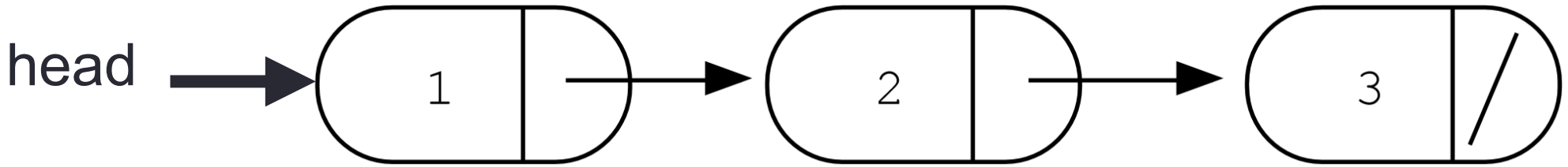
Simplest Linked List (just a head pointer)

- Create an empty list
- Add a node with data “Tina”

```
struct Node {
        string data;
        Node* next;
};
```

Assume the following linked list exists

```
struct Node {
      int data;
      Node *next;
};
```



Evaluate each of the following expressions?

1. `head->data`
2. `head->next->data`
3. `head->next->next->data`
4. `head->next->next->next->data`

A. 1
B. 2
C. 3
D. nullptr
E. Run time error

Printing a list: iterating through a list

Which of the following are valid ways of representing a linked list

A. `Node* head;`
B. `int* head = nullptr;`
C. `Node* head; Node* tail;`
D. Need to define a new type called `LinkedList`

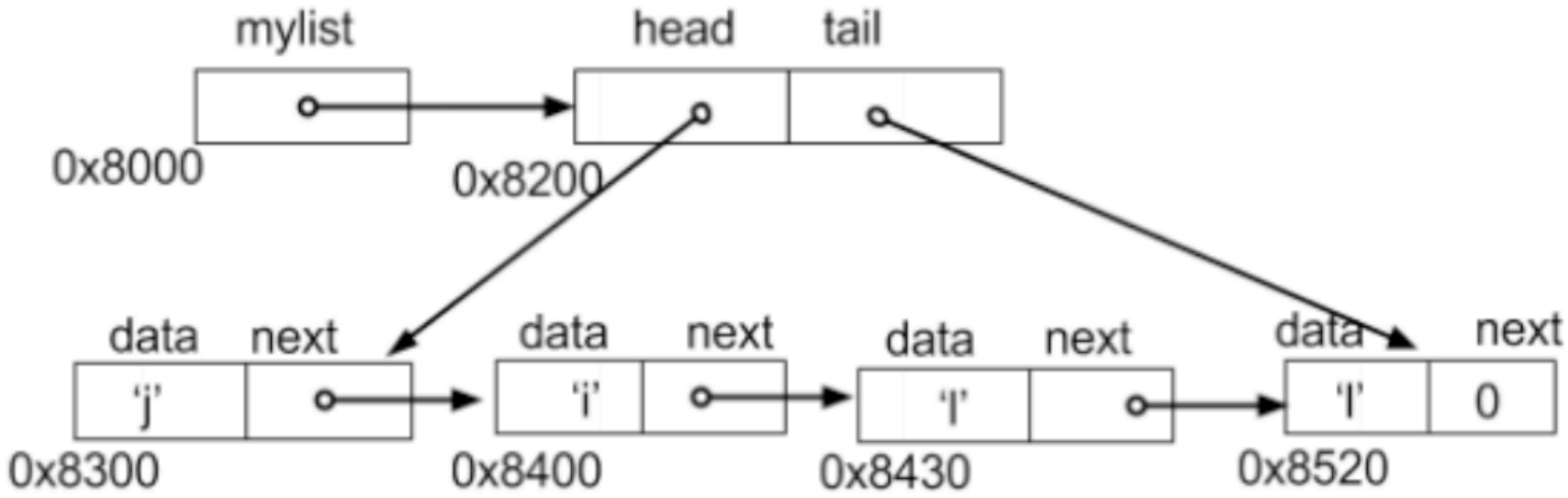
```
struct Node {
      int data;
      Node *next;
};
```

LinkedList datatype

- Define the type LinkedList
- Create an empty list
- Add a node to the list with data “Tina”

```
struct Node {
      string data;
      Node* next;
};
```

Accessing nodes in a linked list



a. **cout<<mylist;**
b. cout<<mylist->tail;
c. cout<<mylist->tail->data;
d. cout<<mylist->head->next;
e. cout<<mylist->head->next->

Next time

- C++ class and Object Oriented Programming