INTRO TO LINKED LISTS

Problem Solving with Computers-II

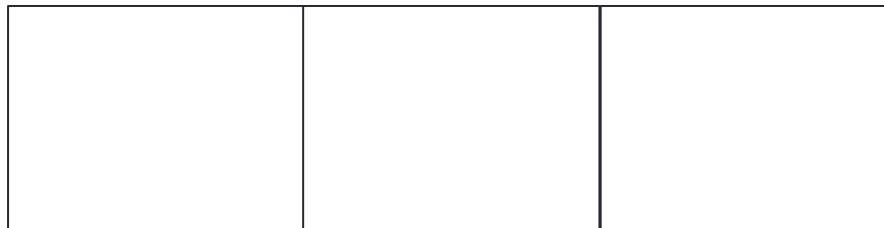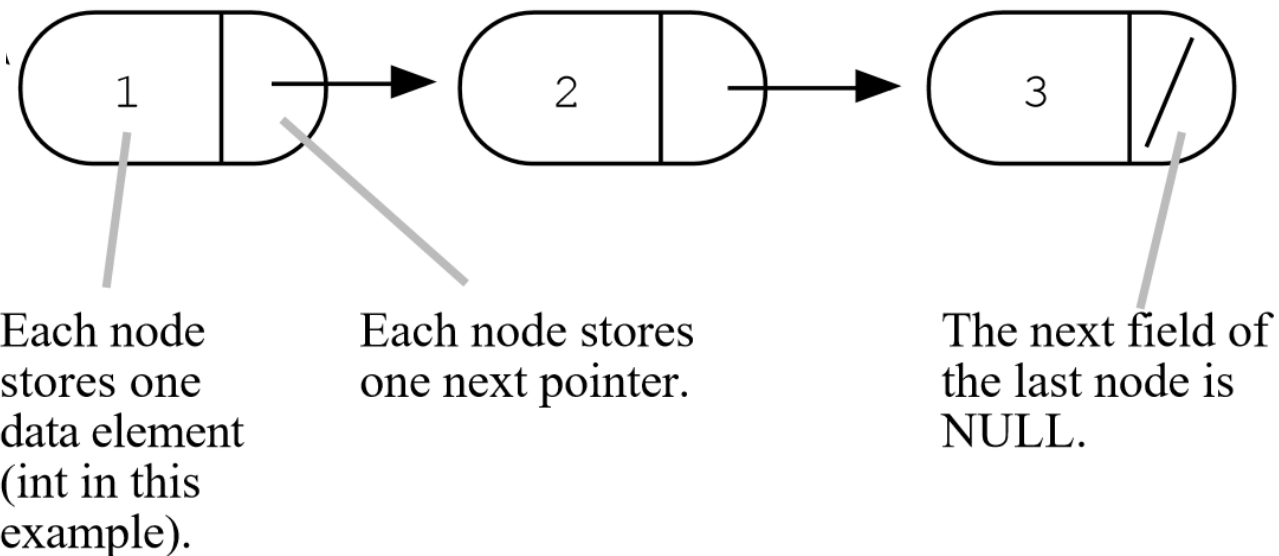# Linked list vs Array

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.

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;
};
```
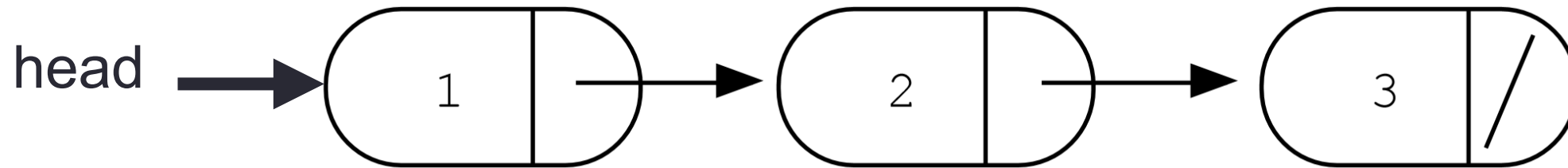
Simplest Linked List (just a head pointer)

- Create an empty list
- Add a node with data "April Sanchez"

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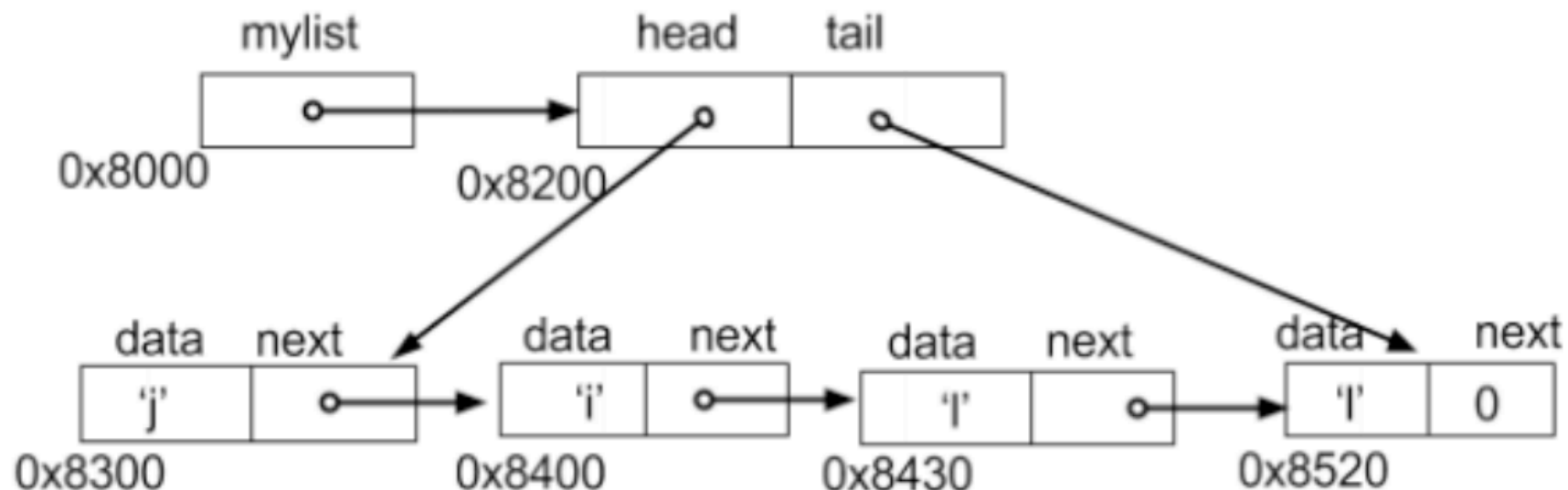
A. 1
B. 2
C. 3
D. nullptr
E. Run time error

LinkedList datatype

- Define the type LinkedList
- Create an empty list
- Add a node to the list with data "April Sanchez"

```
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->

Questions to ask about any data structure:

- **What operations does the data structure support?**

  A linked list supports the following operations:

  1. Insert (a value to the head)
  2. Append (a value to the tail)
  3. Delete (a value)
  4. Search (for a value)
  5. Min
  6. Max
  7. Print all values

- **How do you implement each operation?**
- **How fast is each operation?**

Linked List Abstract Data Type (ADT): OOP style

```cpp
class LinkedList {
public:
    LinkedList();
    ~LinkedList();
    // other public methods


private:
    struct Node {
        int info;
        Node* next;
    };
    Node* head;
    Node* tail;
};
```

Next time

- OOP style Linked List