

# LINKED LISTS

---

Problem Solving with Computers-I

<https://ucsb-cs24-sp17.github.io/>

C++

```
#include <iostream>
using namespace std;

int main(){
    cout<<"Hola Facebook\n";
    return 0;
}
```



# Announcements

- PA3 and PA4 released
- PA3 is due on 5/8
- PA4 is due on 5/15 (a week after that)

# Linked Lists

The Drawing Of List {1, 2, 3}

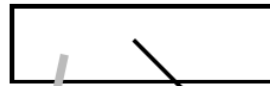


**Array List**

Stack

Heap

head



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

**Linked List**



A “head” pointer local to `BuildOneTwoThree()` keeps the whole list by storing a pointer to the first node.

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.

**What is the key difference between these?**

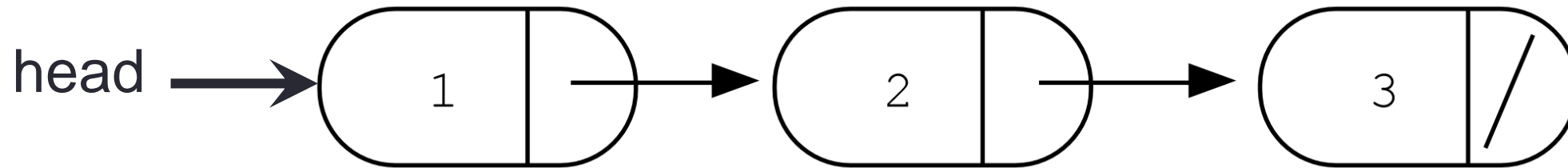
# Create a two node list

- Define an empty list
- Add a node to the list with data = 10

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

# Accessing elements of a list

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



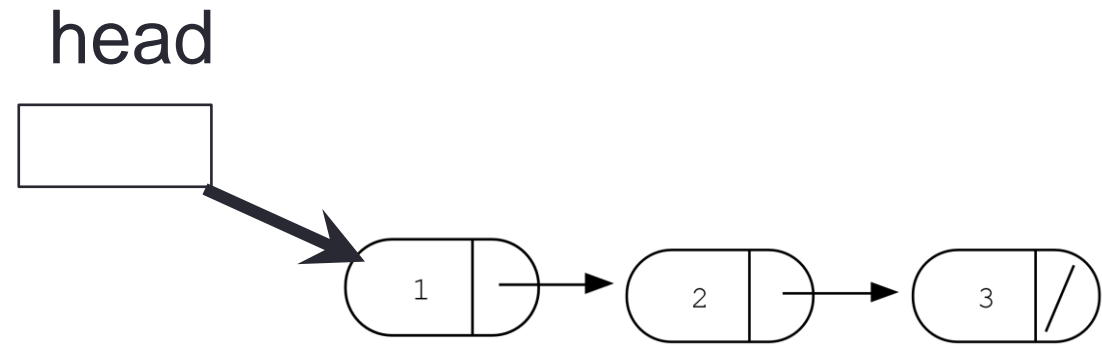
Assume the linked list has already been created, what do the following expressions evaluate to?

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. NULL
- E. Run time error

# Iterating through the list

```
int lengthOfList(Node * head) {  
    /* Find the number of elements in the list */
```



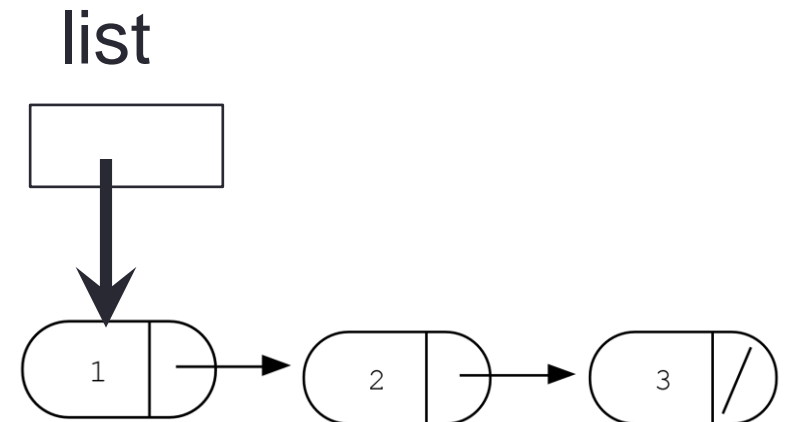
```
}
```

# Linked-list with classes

```
class IntList {
public:
    IntList();          // constructor
    ~IntList();        // destructor
    // other methods
private:
    // definition of Node structure
    struct Node {
        int info;
        Node *next;
    };
    Node *first; // pointer to first node
};
```

# Deleting the list

```
Node* freeLinkedList(Node * list) {  
    /* Free all the memory that was created on the heap*/  
}
```



}



# Next time

- More linked list with classes