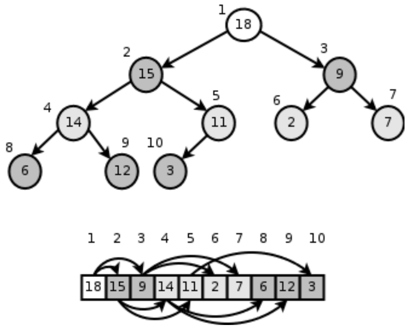
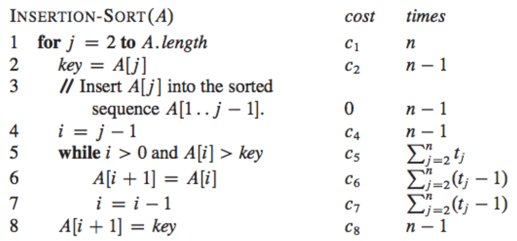
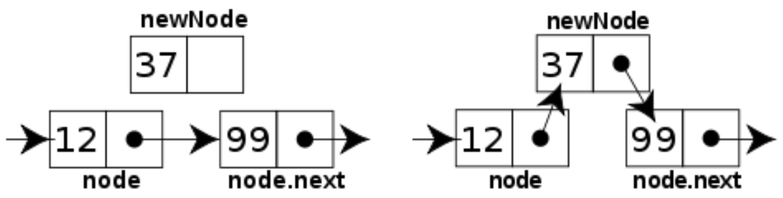




	INSERTION-SORT(A)	*cost*	*times*
1	**for** $j = 2$ **to** $A.length$	c_1	n
2	$key = A[j]$	c_2	$n-1$
3	// Insert $A[j]$ into the sorted sequence $A[1..j-1]$.	0	$n-1$
4	$i = j - 1$	c_4	$n-1$
5	**while** $i > 0$ and $A[i] > key$	c_5	$\sum_{j=2}^{n} t_j$
6	$A[i+1] = A[i]$	c_6	$\sum_{j=2}^{n} (t_j - 1)$
7	$i = i - 1$	c_7	$\sum_{j=2}^{n} (t_j - 1)$
8	$A[i+1] = key$	c_8	$n-1$



WELCOME TO CS 24!

Problem Solving with Computers-II

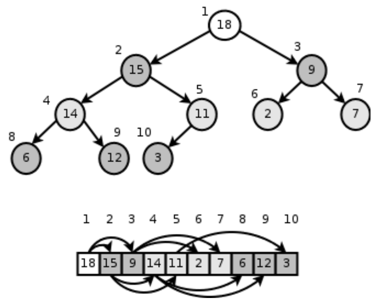
Instructor: Diba Mirza

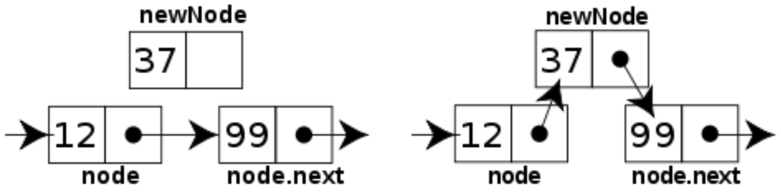
```
#include <iostream>
using namespace std;

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

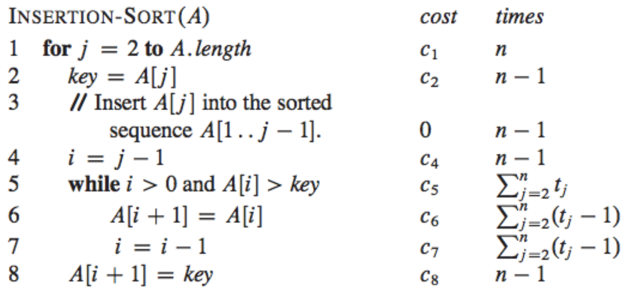
Read the syllabus. Know what's required. Know how to get help.

About this course

- Design and implement **larger programs** that **run fast**
- Organize **data** in programs using **data structures**
- **Analyze** the **complexity** of your programs
- Understand what goes on **under the hood of programs**







INSERTION-SORT(A)		*cost*	*times*
1	**for** $j = 2$ **to** $A.length$	c_1	n
2	$key = A[j]$	c_2	$n-1$
3	// Insert $A[j]$ into the sorted sequence $A[1..j-1]$.	0	$n-1$
4	$i = j - 1$	c_4	$n-1$
5	**while** $i > 0$ and $A[i] > key$	c_5	$\sum_{j=2}^{n} t_j$
6	$A[i+1] = A[i]$	c_6	$\sum_{j=2}^{n} (t_j - 1)$
7	$i = i - 1$	c_7	$\sum_{j=2}^{n} (t_j - 1)$
8	$A[i+1] = key$	c_8	$n-1$

Data Structures and C++   **Complexity Analysis**

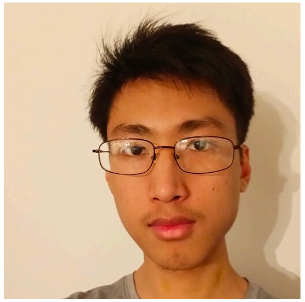
About the team

Diba Mirza

- Prof. Mirza's Office hours: Thurs noon - 1:30p, HFH 1155, or by appointment
- Communication with staff via **Piazza**
- Include [CS24] in the subject line of any email communication with me
- Sections start this week on Thursday
- Office hours start next week

Ask questions about class examples, assignment questions, or other CS topics.

TA Nawal



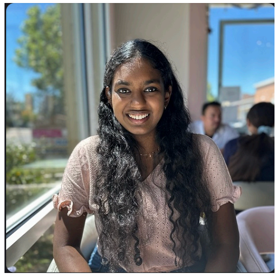
TA Joseph

TA Xinli

TA Yaoyi

LA Zack

LA Ally



LA Sanjana

Course Logistics

- Course website: https://ucsb-cs24.github.io/w24
- Read the syllabus
- If you have a section conflict, you may informally switch your section time.
- No makeup on exams unless its a real emergency!

iClicker Cloud

- Join the class CMPSC24: Problem Solving with Computers-2:

https://join.iclicker.com/GLRN or use the QR code

Recommended textbook

- Problem Solving with C++, Walter Savitch, Edition 9

About lectures

- I will not be a talking textbook
- Ask questions anytime!
- I’ll ask you questions too! Be ready to discuss with the people near you and respond to multiple-choice questions (using the clickers).
- Take a moment to introduce yourself to the people sitting near you.

    - Talk about…
        - your background,
        - experience in CS so far, and
        - what you hope to get out of this class!

About you: When did you take CS16 or an equivalent course?

A. Fall 2023
B. Summer 2023
C. Spring 2023
D. Winter 2023 or earlier

About you…

What is your familiarity/confidence in C++?

A. Know nothing or almost nothing about it.
B. Used it a little, beginner level.
C. Some expertise, lots of gaps though.
D. Lots of expertise, a few gaps.
E. Know too much; I have no life.

About you…

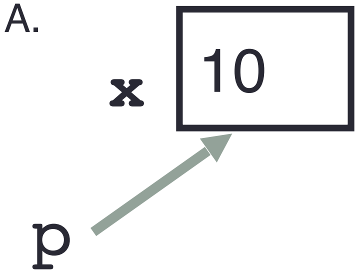
What is your familiarity/confidence with using git or any version control system?

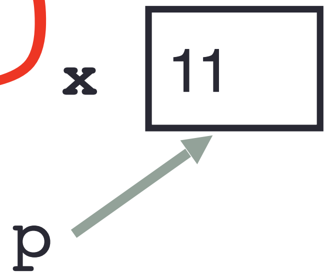
A. Know nothing or almost nothing about it.
B. Used it a little, beginner level.
C. Some expertise, lots of gaps though.
D. Lots of expertise, a few gaps.
E. Know too much; I have no life.

Review: Tracing code involving pointers

```
int* p;
int x = 10;
p = &x;
*p = *p + 1;
```

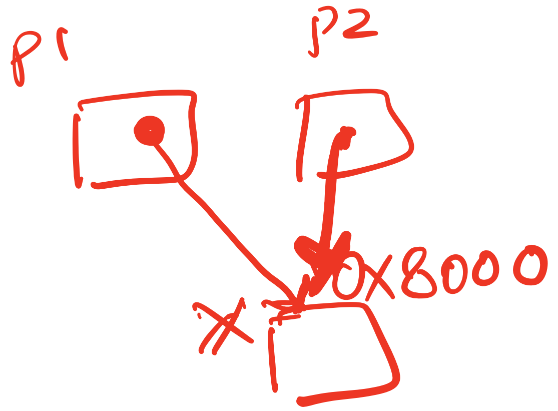
Q: Which of the following pointer diagrams best represents the outcome of the above code?

A.


[✓ B.]


C. Neither, the code is incorrect

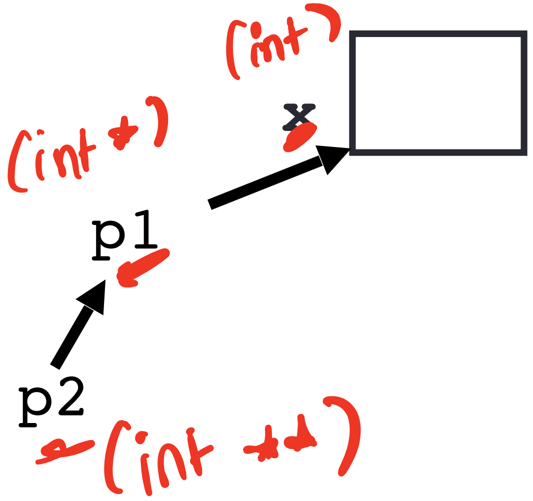
Review: Pointer assignment



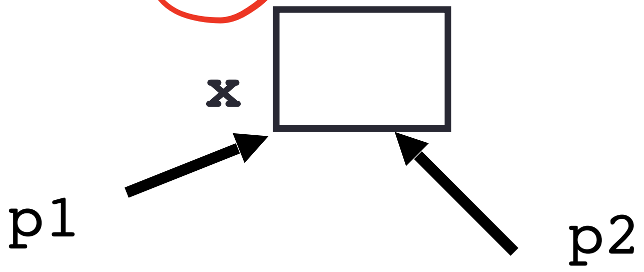
```
int* p1, *p2, x;
p1 = &x;
p2 = p1;
```

Q: Which of the following pointer diagrams best represents the outcome of the above code?

A.



[✓ B.]



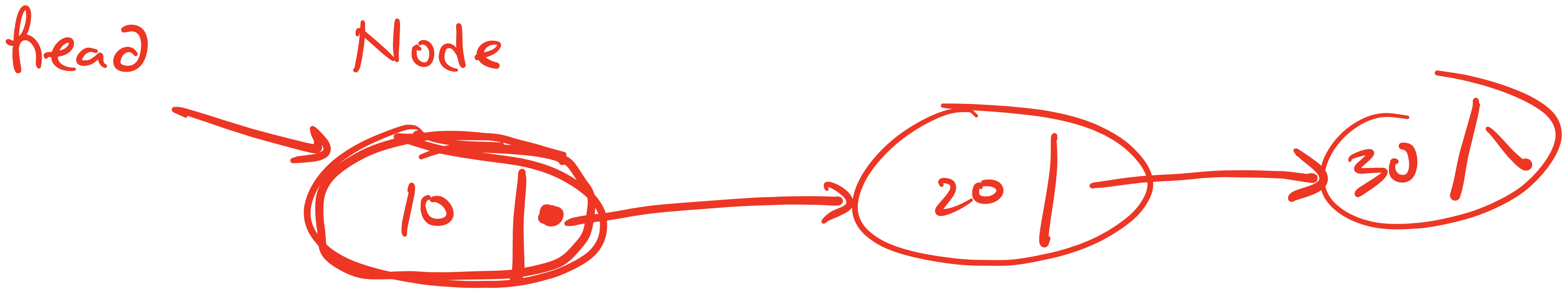
C. Neither, the code is incorrect

Linked list vs Array

Array

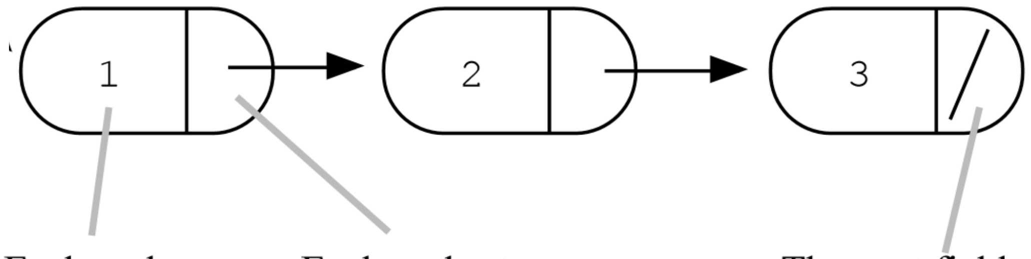
10	20	30

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



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.



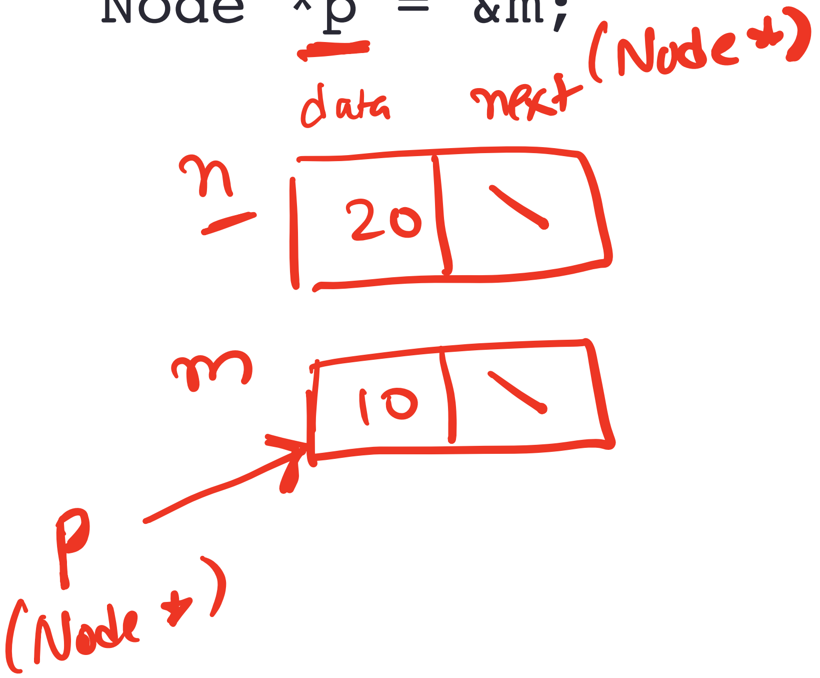
Each node stores one data element (int in this example).

Each node stores one next pointer.

The next field of the last node is NULL.

Review: Accessing structs using pointers

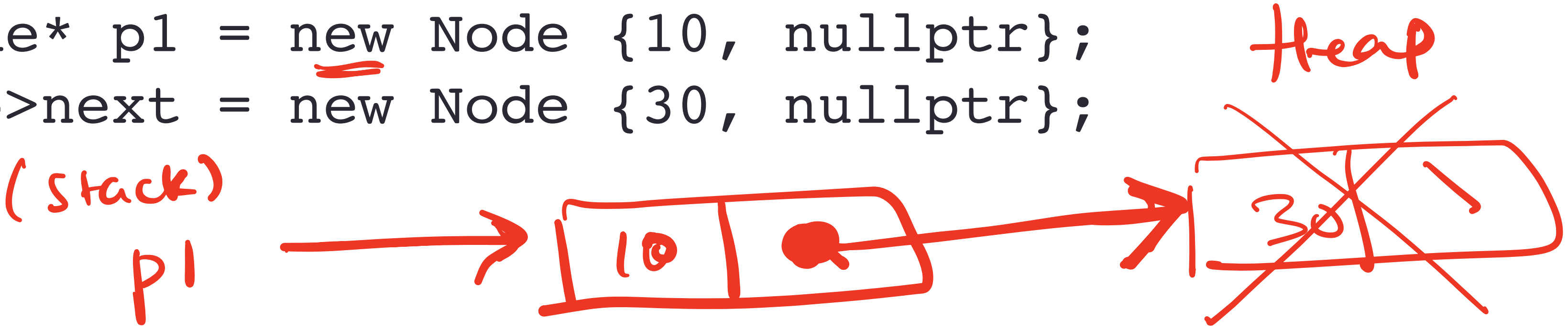
```
Node n {20, nullptr};
Node m {10, nullptr};
Node *p = &m;
```



```
n.data; // 20
n.next; // nullptr
m.data; // 10
(*p).data; // 10
p->data; //10
```

Review: Dynamic memory (new and delete)

```
Node* p1 = new Node {10, nullptr};
p1->next = new Node {30, nullptr};
```



delete p1→next;

p1→next = nullptr;

A. p1

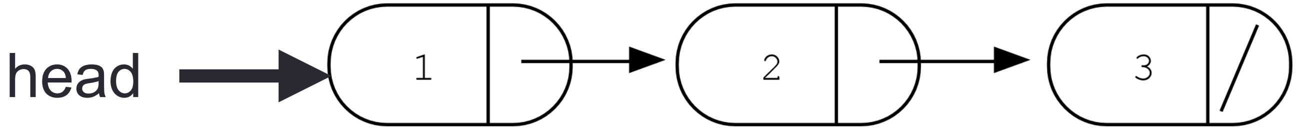
B. Node with value 10

[✓ C.] Node with value 30

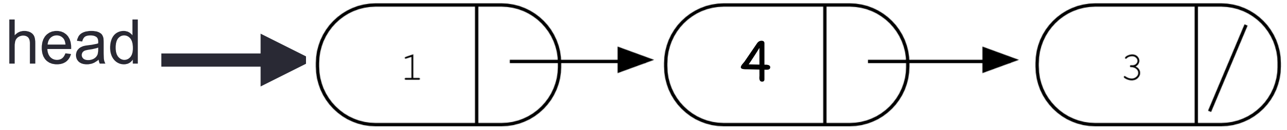
How does the given code modify the provided linked list?

```
Node* p = head;
p = p->next;
p->data = 4;
```

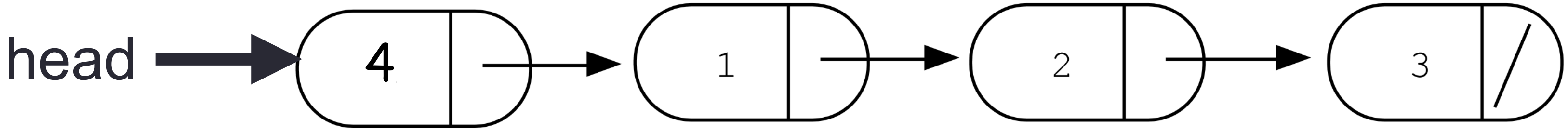
```
struct Node {
        int data;
        Node* next;
};
```



A.



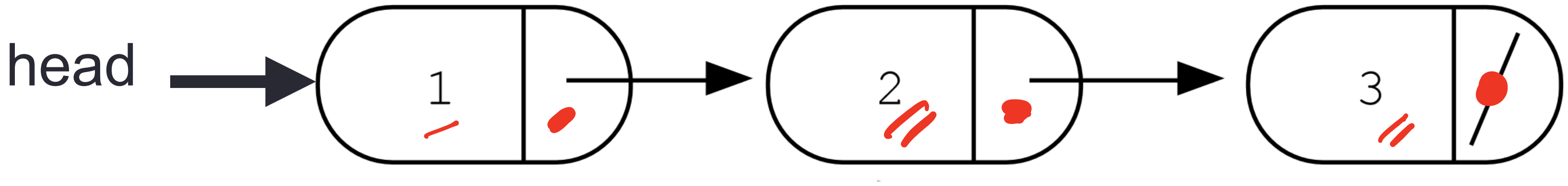
B.



C. Something else

Assume the following linked list exists

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



Evaluate each of the following expressions?

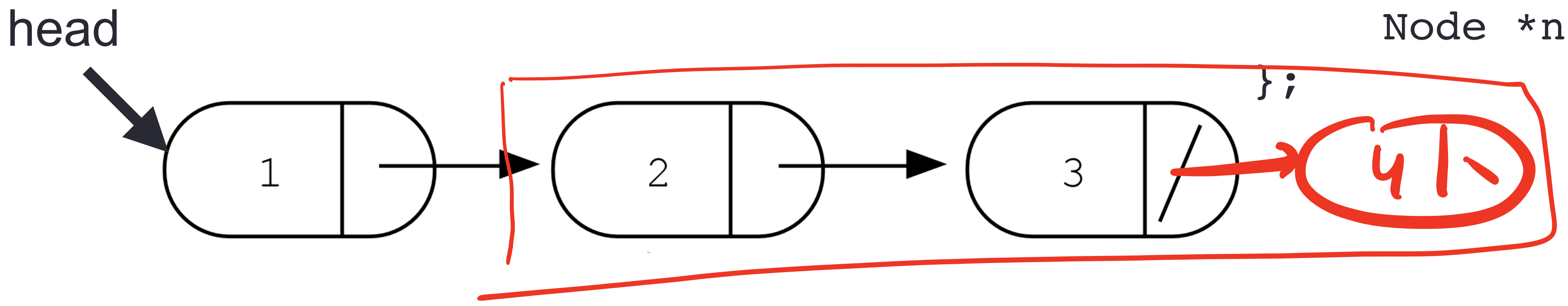
```
1. head->data   // 1
2. head->next->data
3. head->next->next->data
4. head->next->next->next->data
```

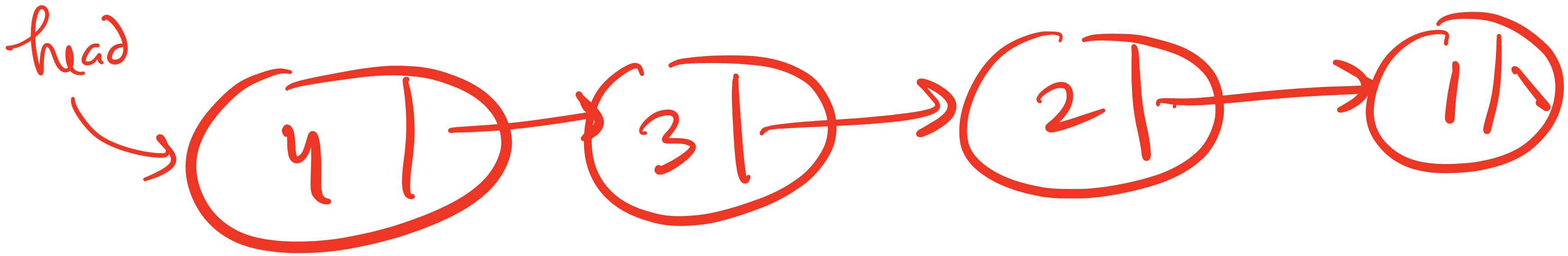
nullptr

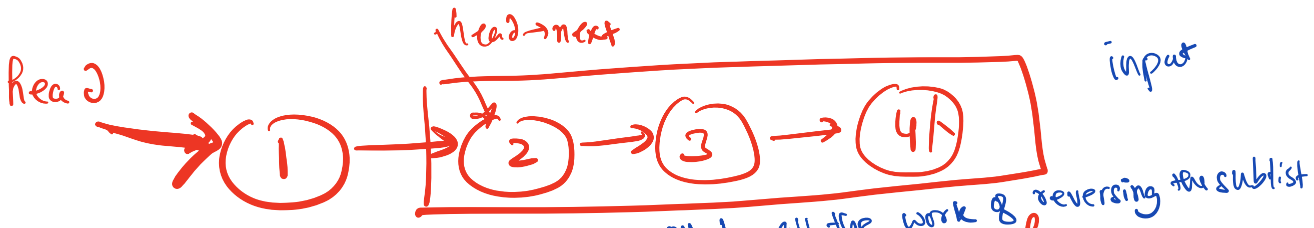
A. 1
B. 2
C. 3
D. nullptr
[✓ E.] Run time error

Write a C++ function to reverse a singly linked list

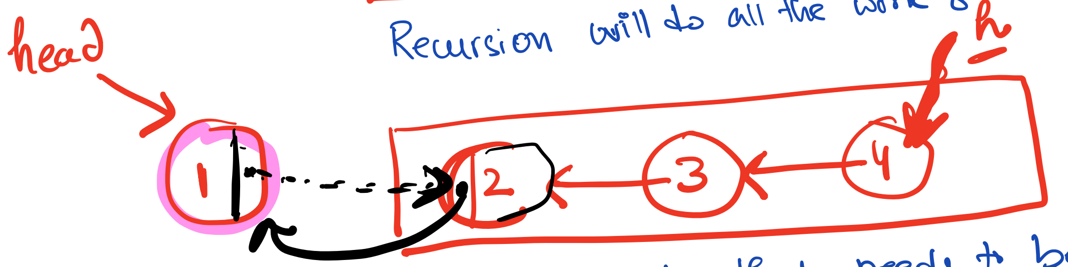
```
struct Node {
        int data;
        Node *next;
};
```



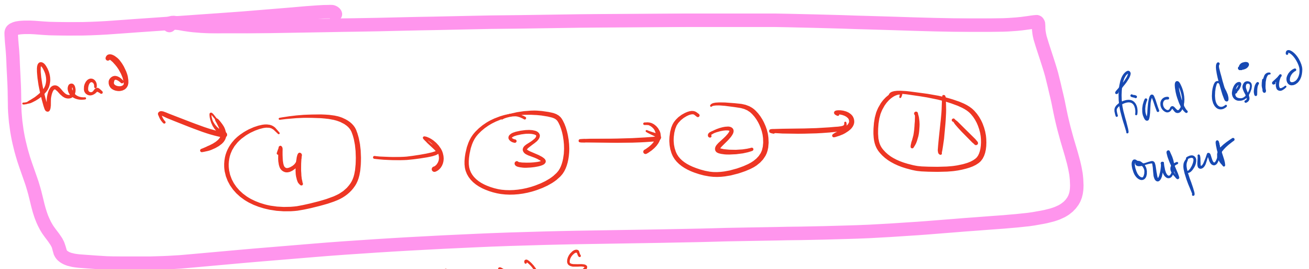




Recursion will do all the work & reversing the sublist



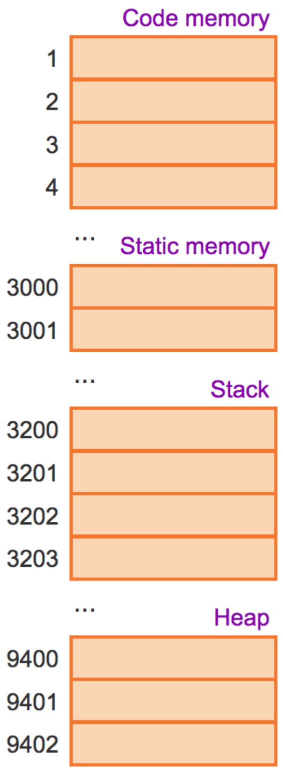
What is the extra work that needs to be done to finish the task? Connect node 2's next to node 1
Set node 1's next to nullptr



```
Node * reverse( Node * head) {
    if (!head) return nullptr;              // Base cases
    if (!head->next) return head;
 h = reverse (head -> next);  // reverses the sublist 2→3→4
 head -> next -> next = head;   // Extra work to finish the task
  head ->next = nullptr            beyond what the recursive call
   return h;                        accomplishes.
}
```

Review: C++ Program's Memory Regions

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

// Program is stored in code memory

int myGlobal = 33;     // In static memory

void MyFct() {
   int myLocal;          // On stack
   myLocal = 999;
   cout << " " << myLocal;
}

int main() {
   int myInt;               // On stack
   int* myPtr = nullptr; // On stack
   myInt = 555;

   myPtr = new int;         // In heap
   *myPtr = 222;
   cout << *myPtr << " " << myInt;
   delete myPtr; // Deallocated from heap

   MyFct(); // Stack grows, then shrinks

   return 0;
}
```



Which of the following is true about data created on the **heap** region of memory?

A. Stores the local variables of a function

B. Stores global variables

[✓ C.] Any data created on the heap stays there FOREVER or until the programmer explicitly deletes it

The code regions store program instructions. myGlobal is a global variable and is stored in the static memory region. Code and static regions last for the entire program execution.

Two important facts about Pointers

1) A pointer can only point to one type –(basic or derived) such as `int`, `char`, a `struct`, a class another pointer, etc

2) After declaring a pointer: `int *ptr;`

   `ptr` doesn't actually point to anything yet.

   We can either:

   - ➢make it point to something that already exists, OR
   - ➢allocate room in memory for something new that it will point to

Next time

- Abstract Data Types