REVIEW POINTERS LINKED LISTS

Problem Solving with Computers-II

Have you implemented a linked-list before?

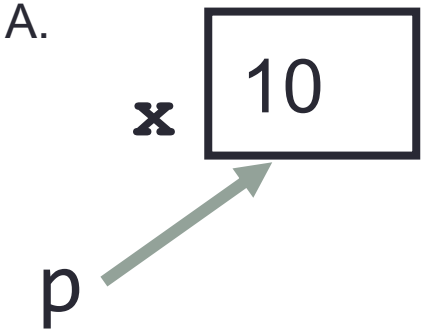
A. Yes
B. No

Tracing code involving pointers

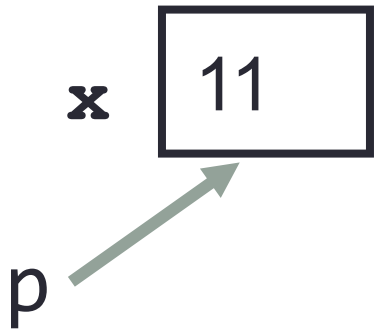
```
int *p, x = 10;
p = &x;
*p = *p + 1;
```

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

A.



B.



C. Neither, the code is incorrect

Pointers

- Pointer: A variable that contains the address of another variable
- Declaration: *type* * pointer_name;

```
int *p;
```

p is an uninitialized pointer.
What is outcome of doing the following?
cout<<*p;

How do we initialize a pointer?

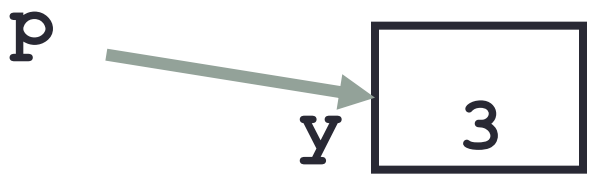
How to make a pointer point to something

```
int *p;
int y;
```


```
p = &y;
```

To access the location of a variable, use the address operator ‘&’

Two ways of changing the value of a variable



Change the value of y directly:

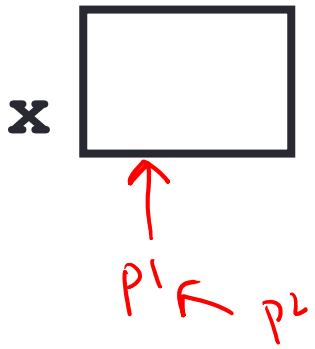
Change the value of y indirectly (via pointer p):

Pointer assignment

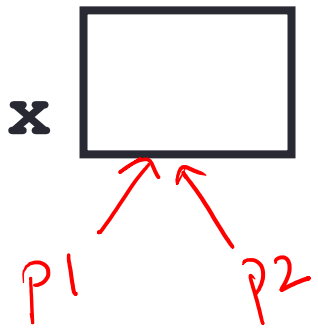
```
int *p1, *p2, x;
p1 = &x;
p2 = p1;
```

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

A.



B.



C. Neither, the code is incorrect

Dynamic memory allocation

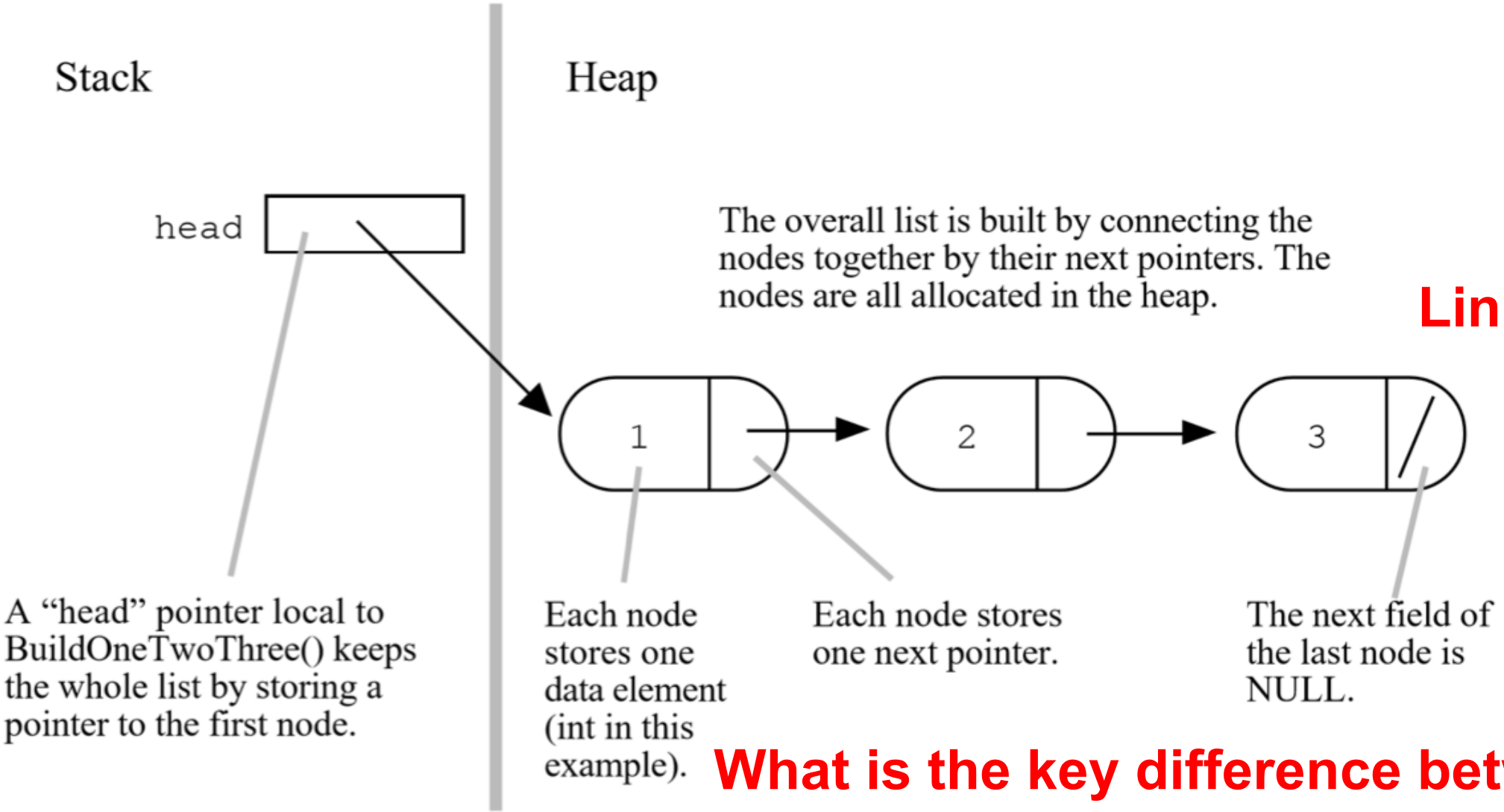
- To allocate memory on the heap use the ‘new’ operator
- To free the memory use delete

```
int *p= new int;
delete p;
```

Linked Lists

Array List

The Drawing Of List {1, 2, 3}

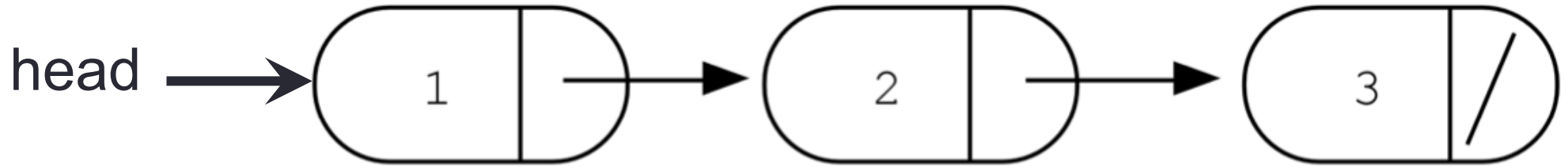


Linked List

What is the key difference between these?

Accessing elements of a list

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



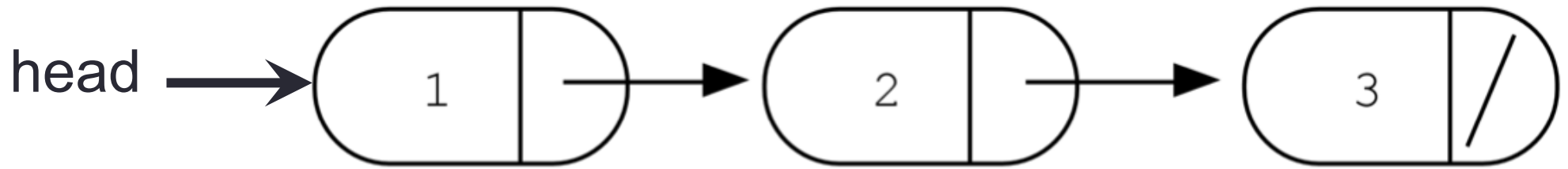
Assume the linked list has already been created, what do the following expressions evaluate to?

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

Working with pointers to structs

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



Create a two node list

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

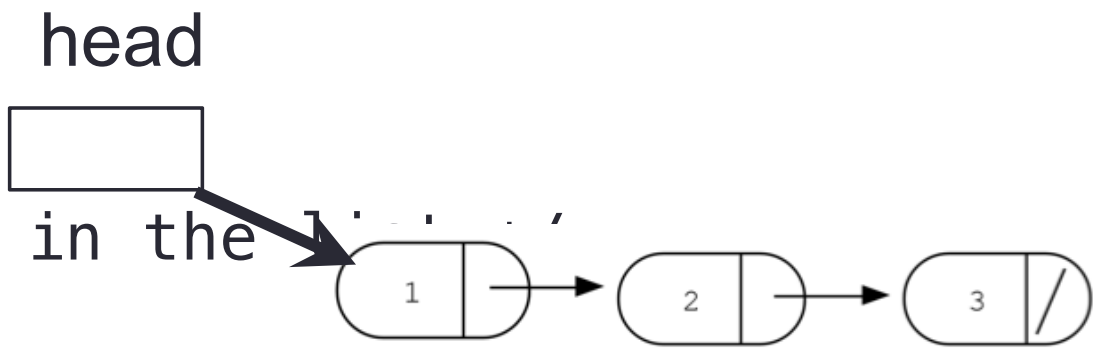
```
struct Node {
      int data;
      Node *next;
};
```

Iterating through the list

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




}
```



Linked-list with classes

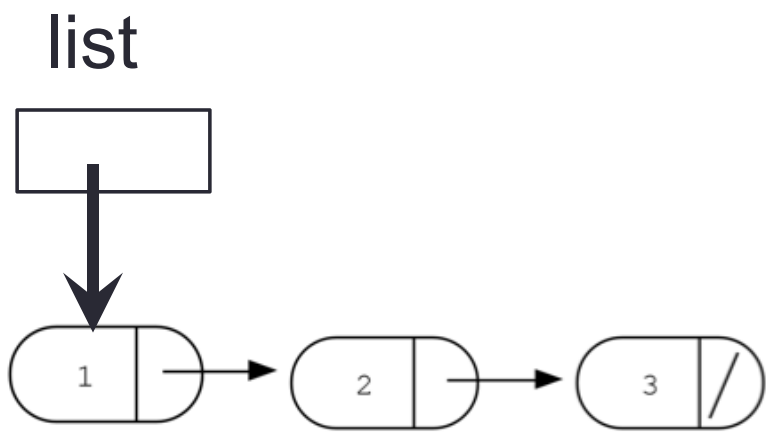
```
class IntList {
public:
    IntList();              // constructor
    ~IntList();              // destructor
    // other methods
private:
    // definition of Node structure
    struct Node {
        int info;
        Node *next;
    };
    Node *first; // pointer to first node
};
```

Deleting the list

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




}
```



Next time

- More linked list with classes