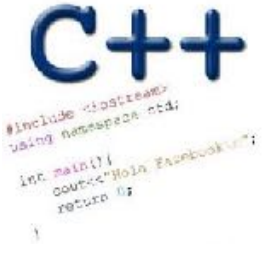
MORE LINKED LISTS

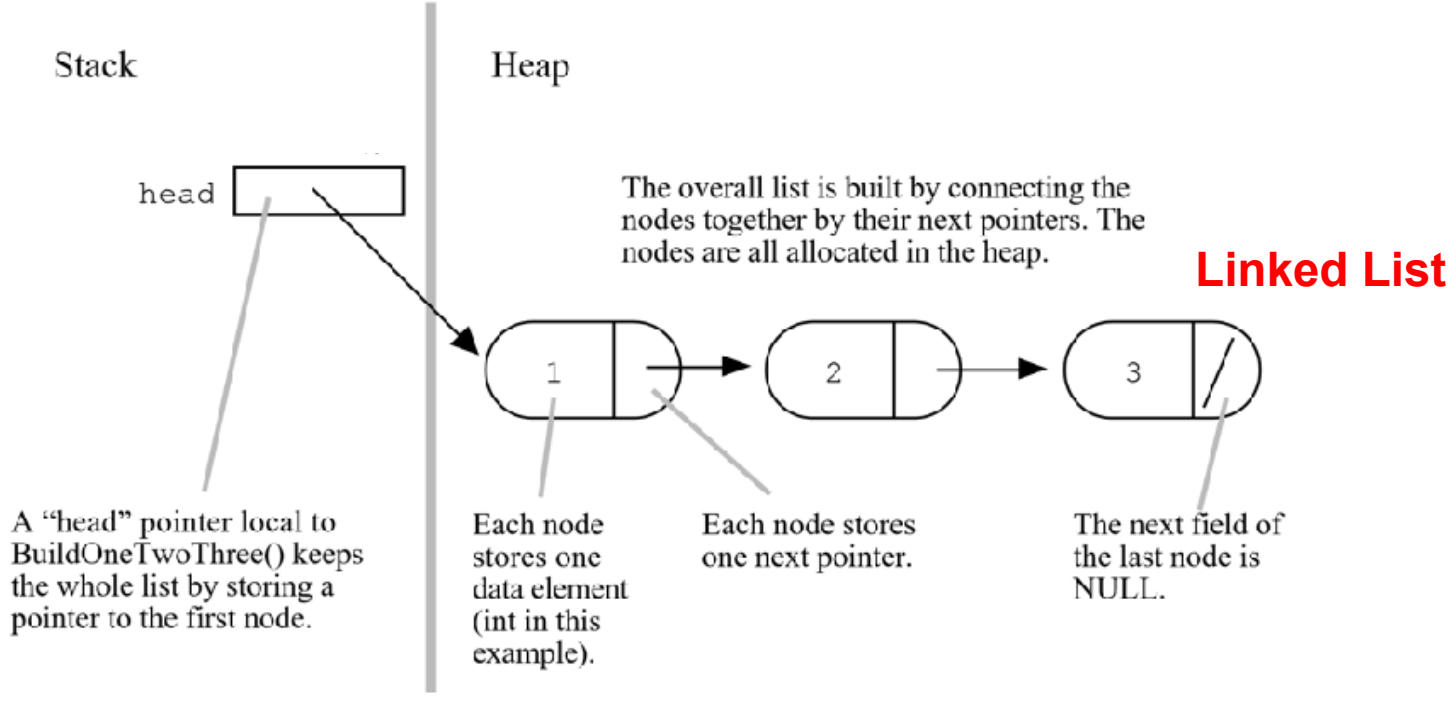
Problem Solving with Computers-I



Linked Lists

Array List

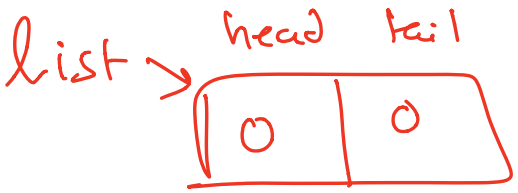
The Drawing Of List {1, 2, 3}

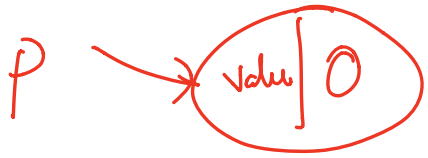


Linked List

A. Yes

B. No





Would this code correctly append a node to an empty Linked List?

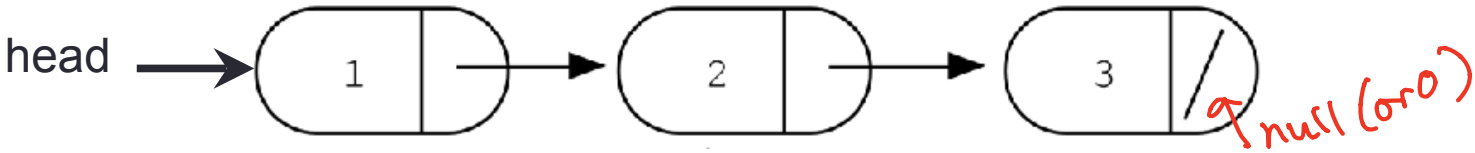
```
Node *p = new Node;
p->data = value;
p->next = 0;
list->tail->next = p;
list->tail = p;
if (list->head == NULL){

    list->head = p;
} else {
    list->tail->next = p;

}list->tail = p;
```

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

dereferencing a null pointer

(seg fault)

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

Creating a small list

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

```
LinkedList mylist;
mylist.head = 0;
mylist.tail = 0;
```
} empty list on the stack

```
Node *p = new Node;
p->data = 10
p->next = 0
list.head = p; list.tail = p;
```

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

```
struct LinkedList {
    Node *head;
    Node *tail;
};
```

Inserting a node in a linked list

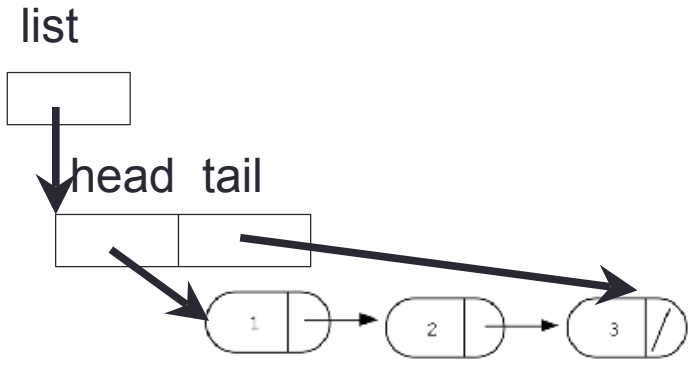
```
Void insertToHeadOfList(LinkedList* h, int value) ;
```

Code available on github

Iterating through the list

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

Code on github

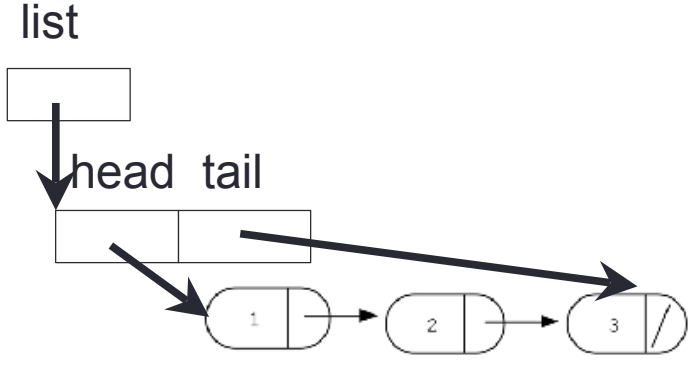



```
}
```

Deleting the list

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

Code on github.




```
}
```

Q: Which of the following functions returns a dangling pointer?

```
int* f1(int num){
     int *mem1 =new int[num];
     return(mem1);
}
```

```
int* f2(int num){
      int mem2[num];
      return(mem2);
}
```

A. f1

B. f2

C. Both

Next time

- More linked lists
- Dynamic arrays
- Dynamic memory pitfall