

# DYNAMIC MEMORY ALLOCATION

## LINKED LISTS

---

Problem Solving with Computers-I

C++

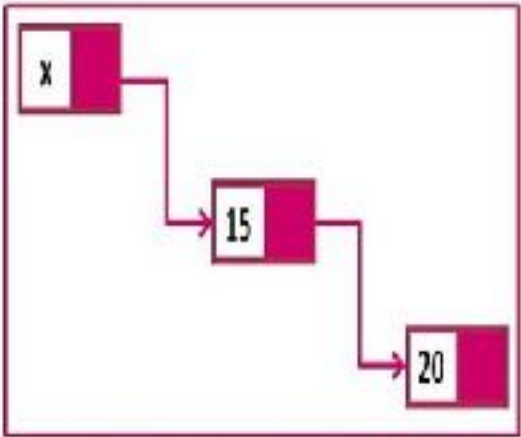
```
#include <iostream>
using namespace std;

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

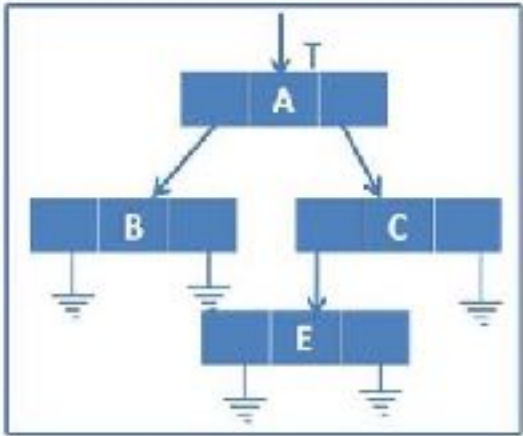


# Different ways of organizing data!

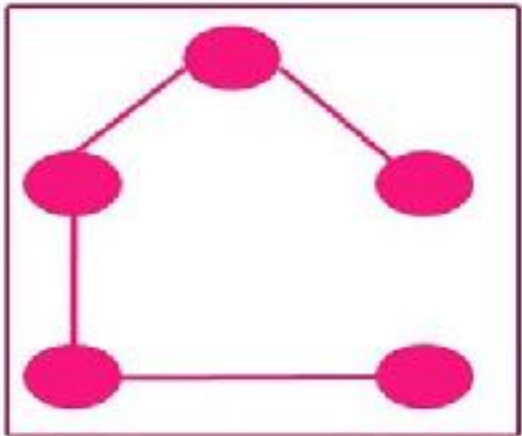
15	20	30
----	----	----



Link list

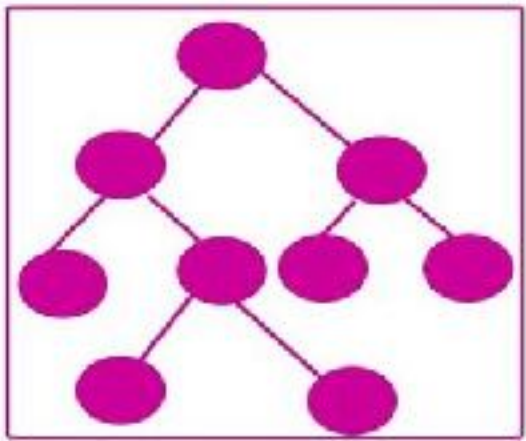


list



spanning tree

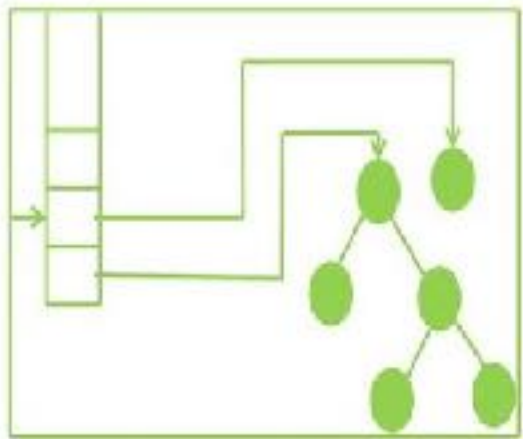
Array List



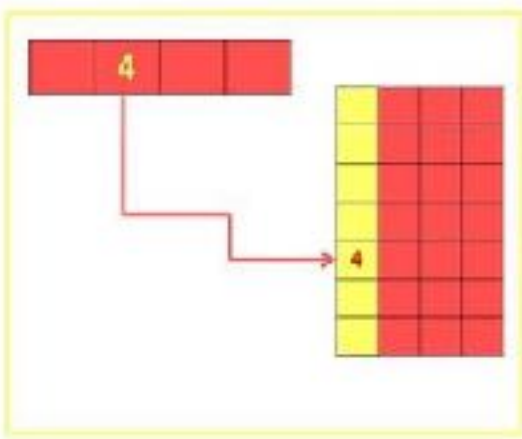
Tree



Graph



Stack



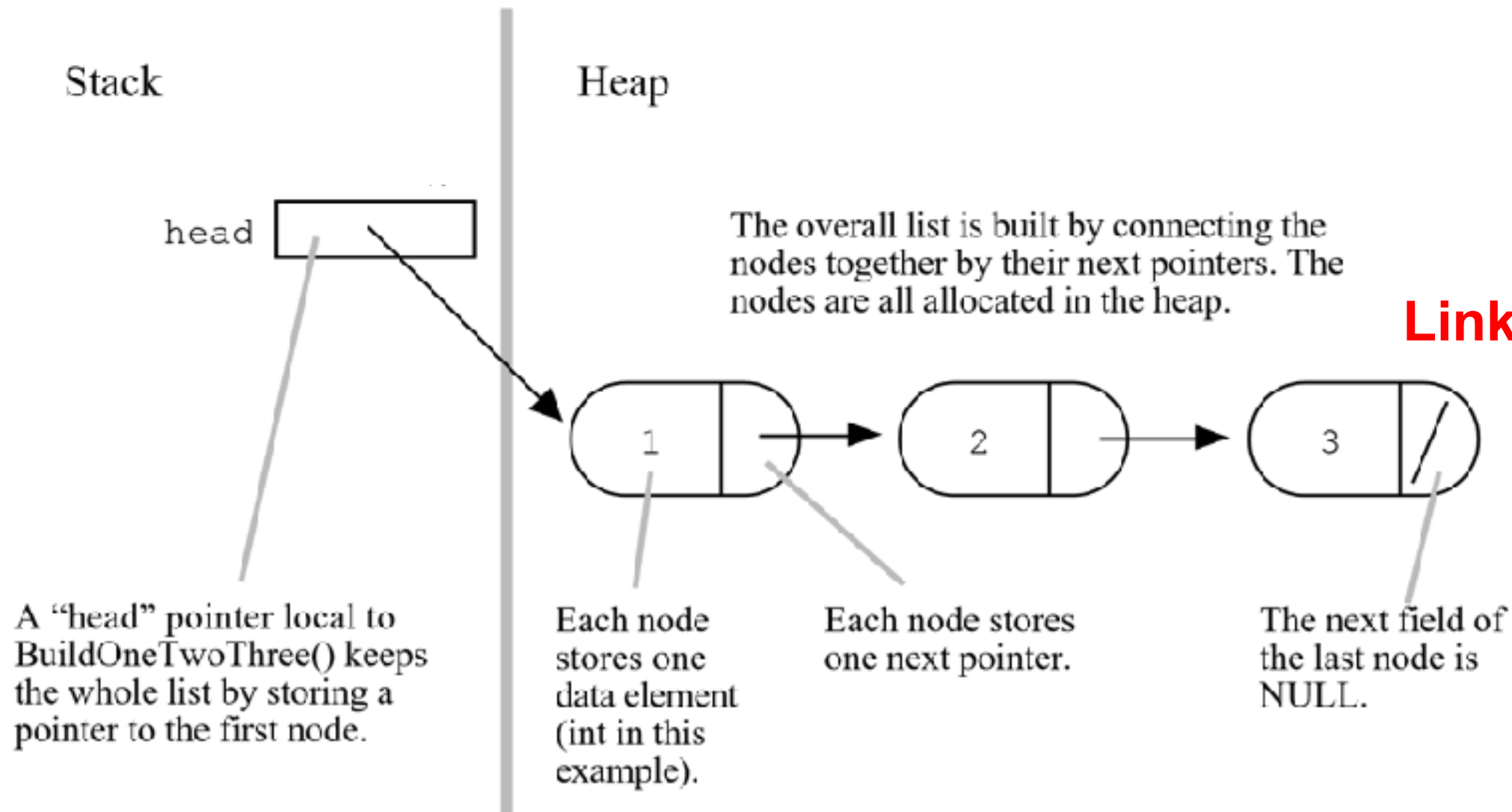
Hashing

# Linked Lists

The Drawing Of List {1, 2, 3}

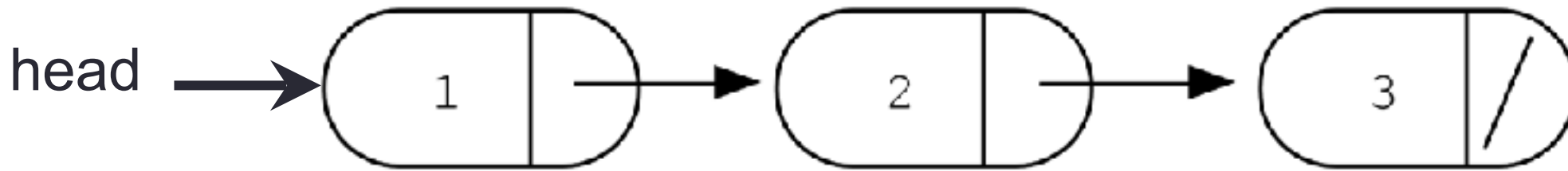
1	2	3
---	---	---

Array List



# 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

# Creating a small list

- Define an empty list
- Add a node to the list with data = 3
- Add a node to the list with data = 2
- Add a node to the list with data = 1

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

# Inserting a node in a linked list

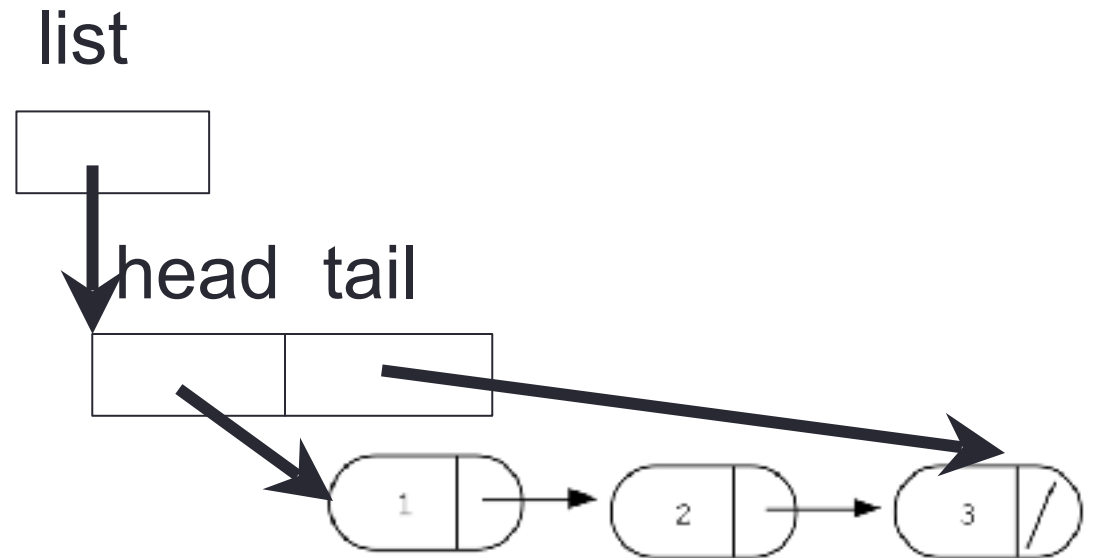
```
void insertToHeadOfList(LinkedList* h, int value) ;
```

# Building a list from an array

```
LinkedList * arrayToLinkedList(int a[], int size);
```

# Iterating through the list

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





# Deleting the list

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

