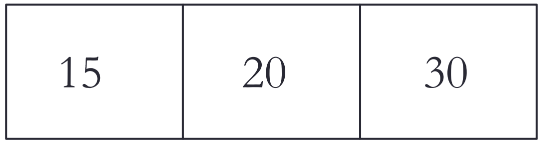
DYNAMIC MEMORY ALLOCATION
LINKED LISTS

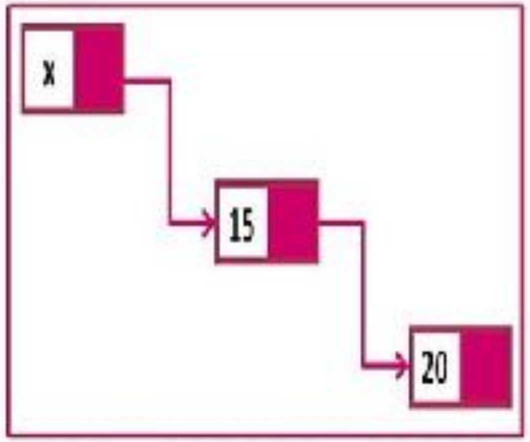
Problem Solving with Computers-I

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

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

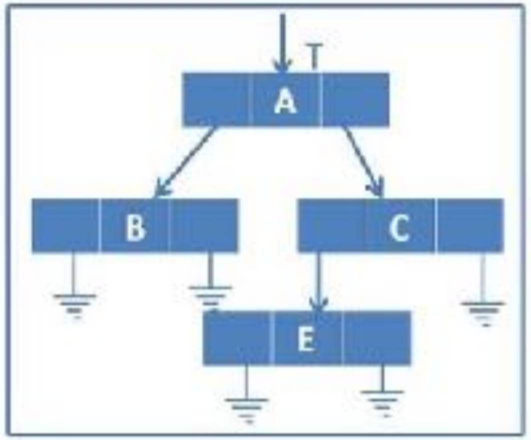
Different ways of organizing data!



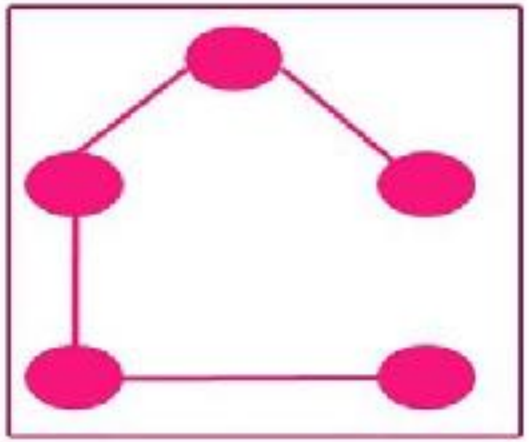
Array List



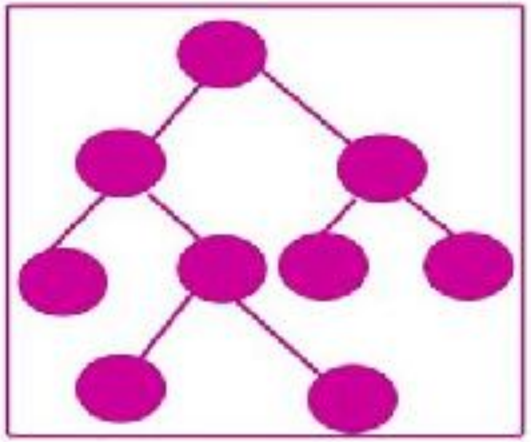
Link list



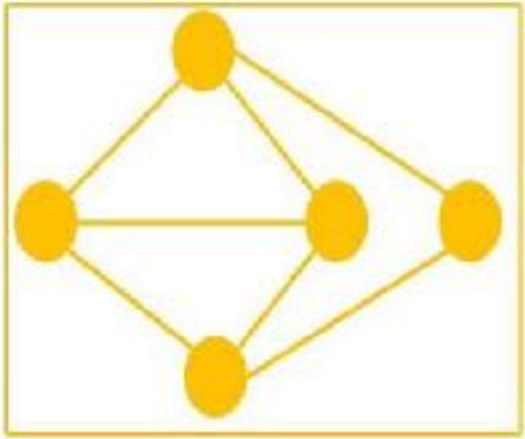
list



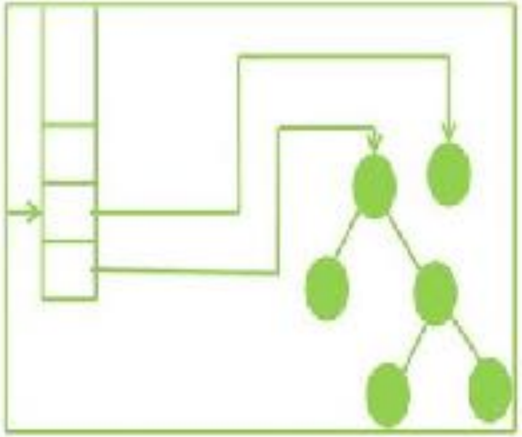
spanning tree



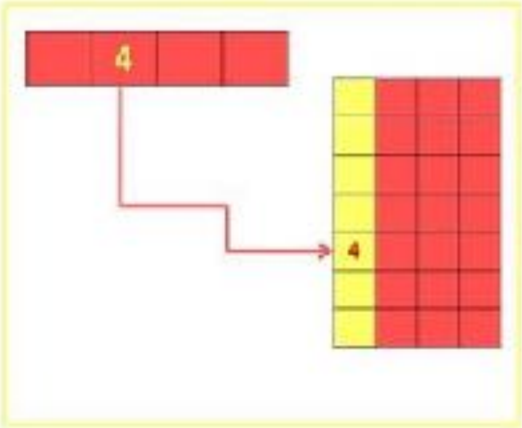
Tree



Graph



Stack



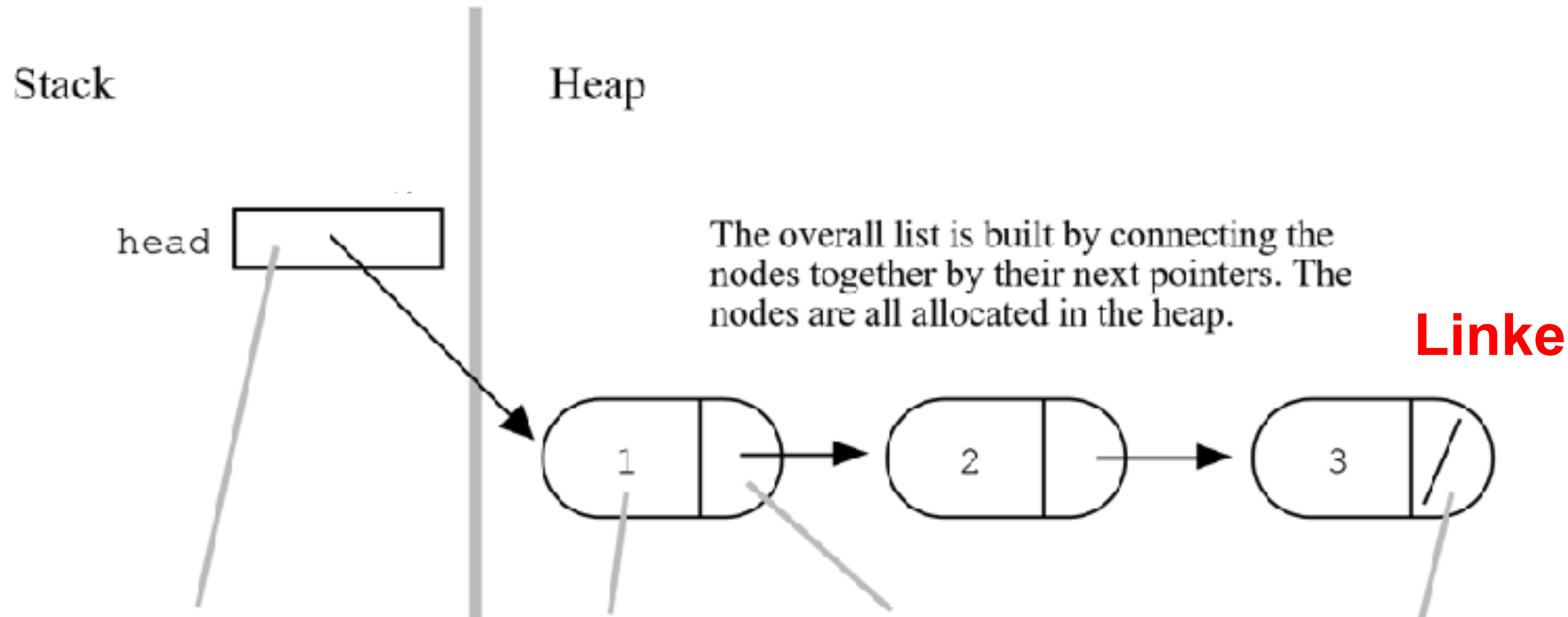
Hashing

Linked Lists

The Drawing Of List {1, 2, 3}

1	2	3

Array List



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.

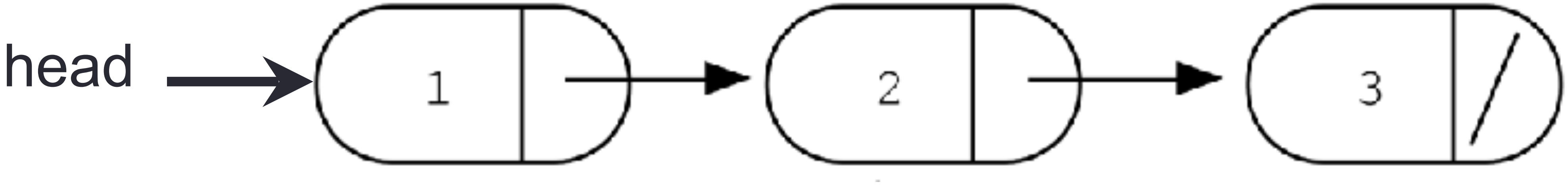
Creating a small list

- Define an empty list
- Add a node to the list with data = 10
- Add a second node with data = 20

```
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 A
2. head->next->data B
3. head->next->next->data C
4. head->next->next->next->data E

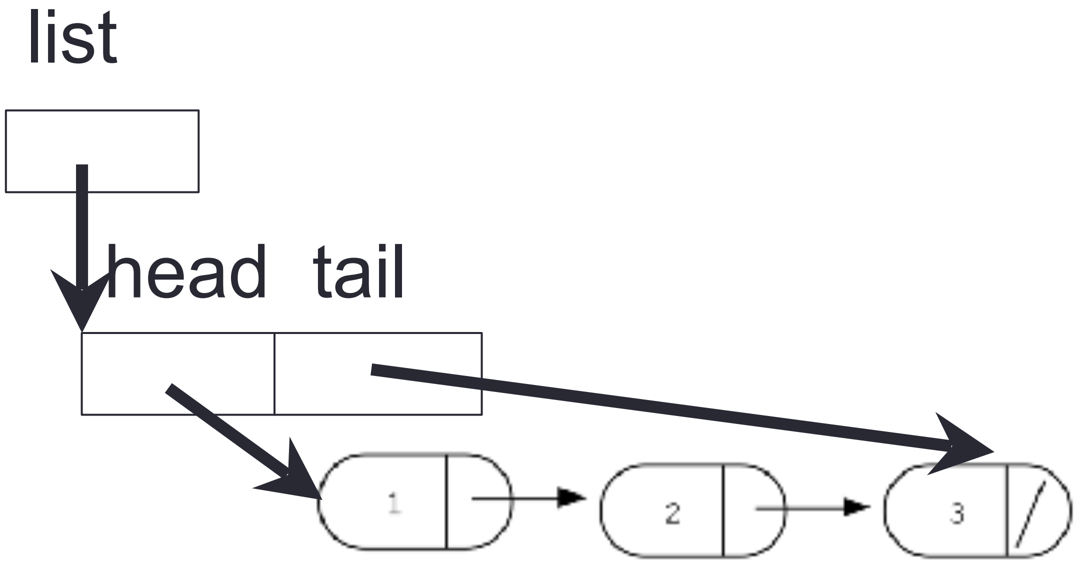
A. 1
B. 2
C. 3
D. NULL
E. Run time error

Inserting a node in a linked list

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

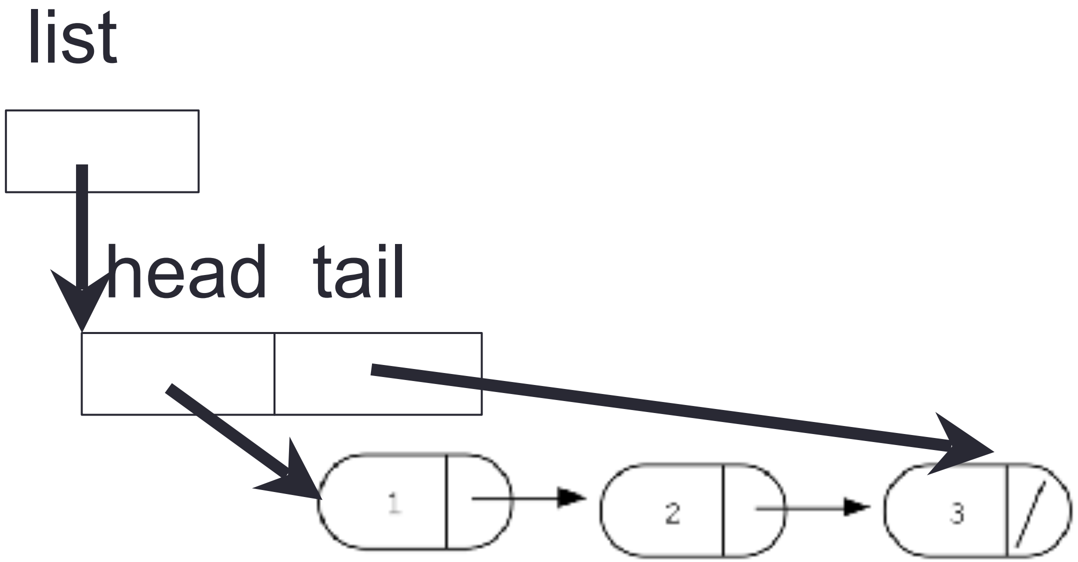
Iterating through the list

```
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*/












}
```


list
head
tail
1
2
3