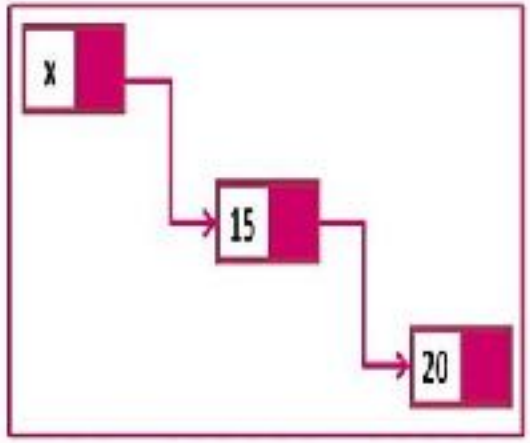
DYNAMIC MEMORY ALLOCATION LINKED LISTS

Problem Solving with Computers-I

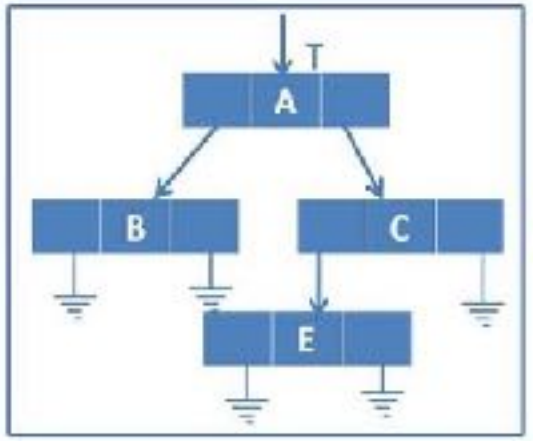
Different ways of organizing data!

15	20	30

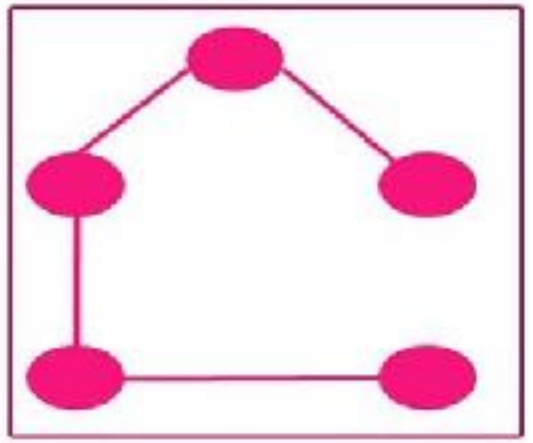
Array List



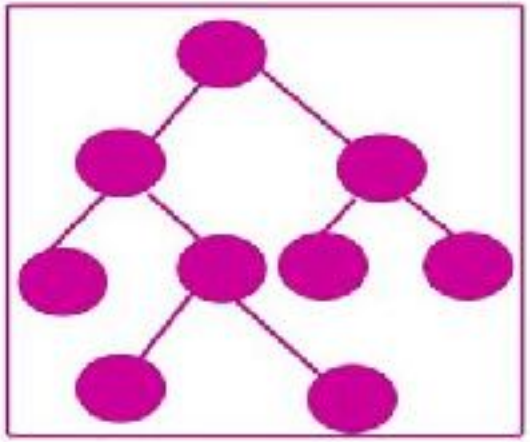
Link list



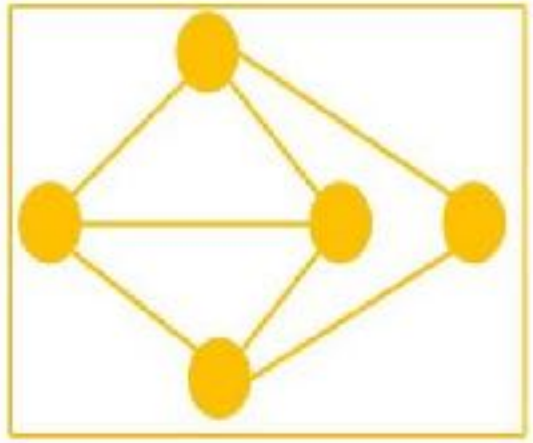
list



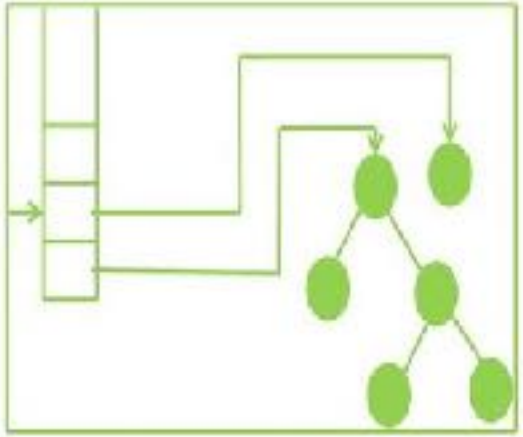
spanning tree



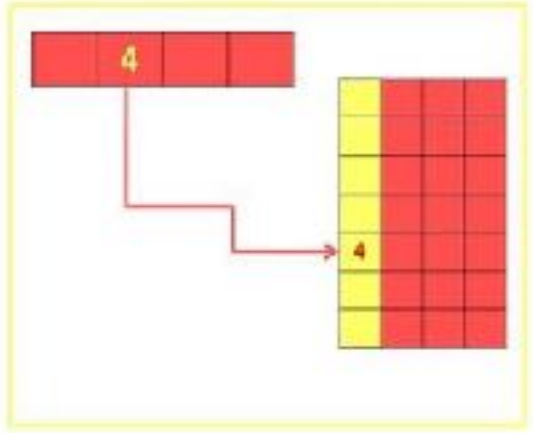
Tree



Graph



Stack

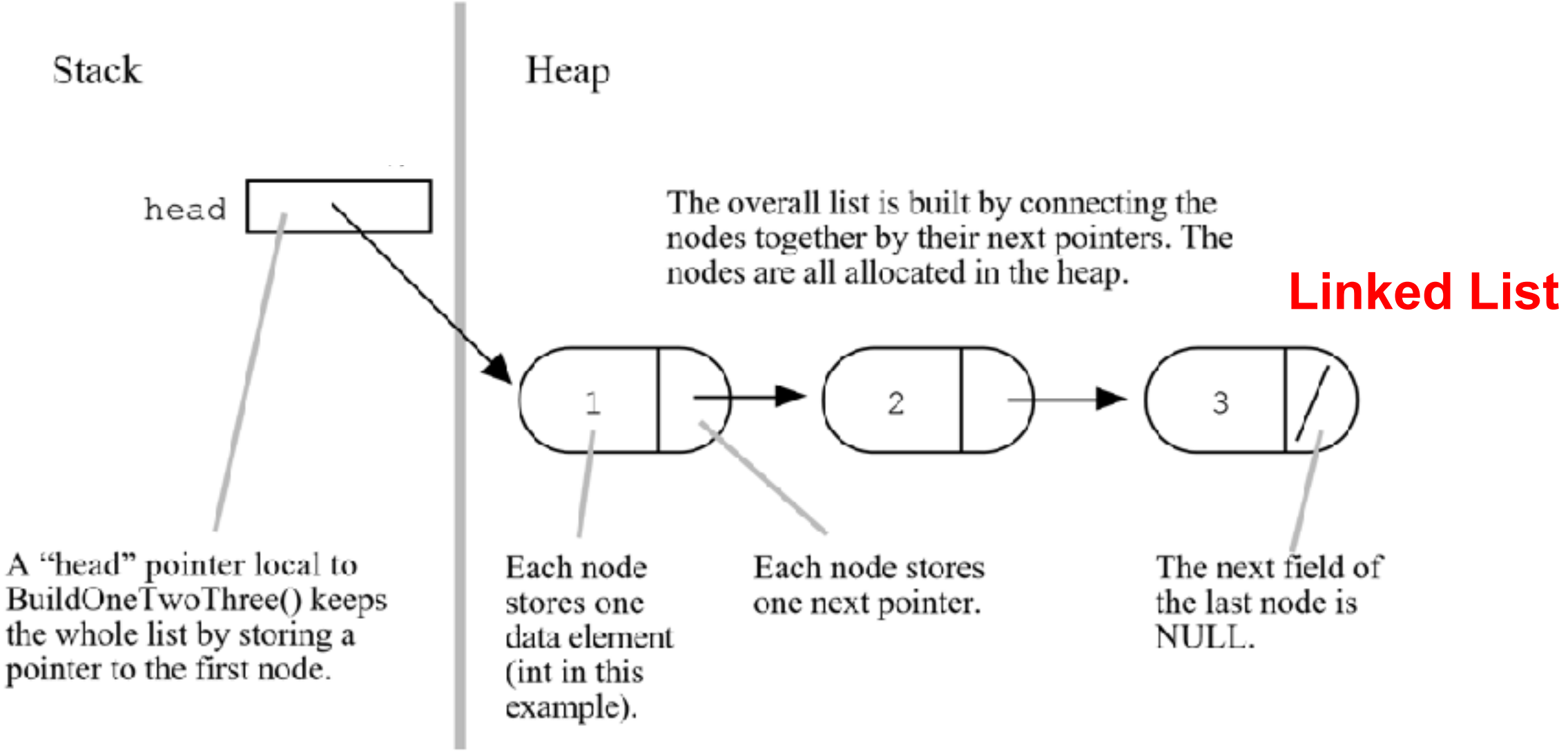


Hashing

Linked Lists

Array List

The Drawing Of List {1, 2, 3}



Linked List

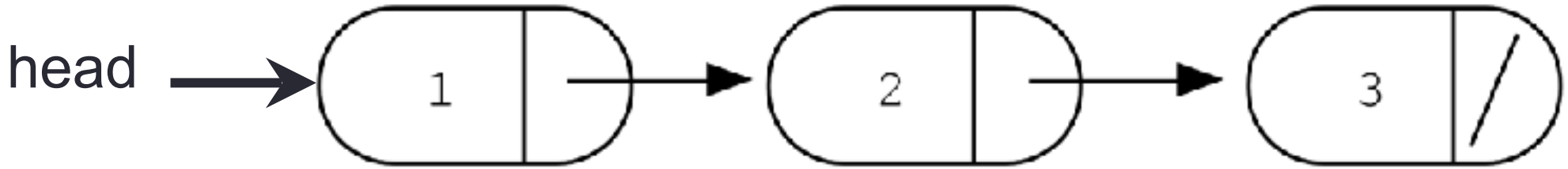
Creating a small list

- 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
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

Inserting a node in a linked list

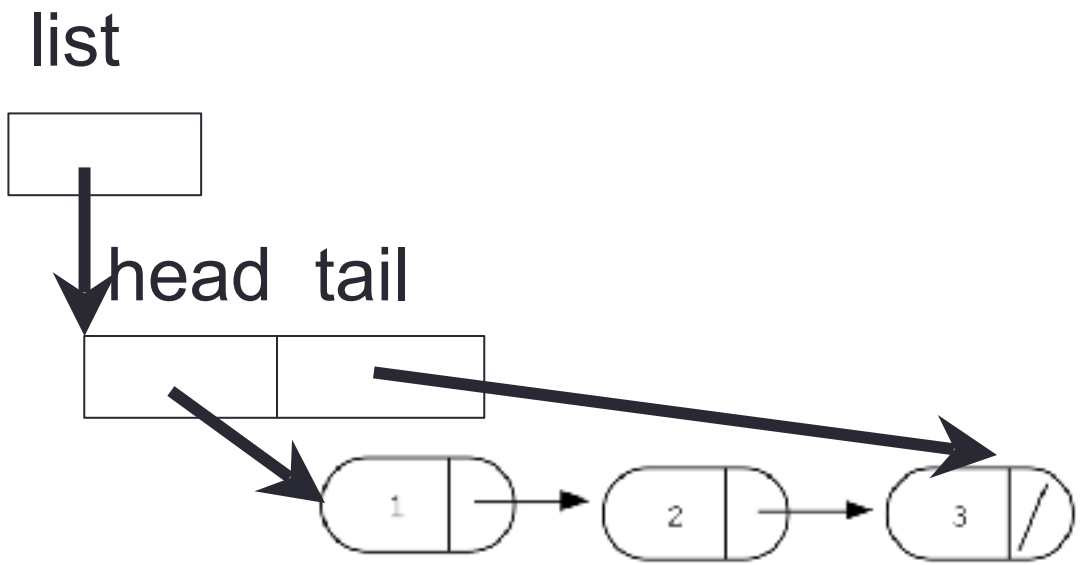
```
Void insertToHeadOfList(LinkedList* h, int value) ;
```

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*/
}
```