This is CS50

Section, Week 10

TA: Andi Peng

Agenda

- Helpful Tips
- Review of Topics Sent In
- Quiz 1 Practice Problems
- Questions?
- Pump Up Speech Part II

Topics (non-exhaustive)

- stacks
- queues
- linked lists
- hash tables
- trees
- Huffman Coding
- tries
- TCP/IP
- HTTP
- HTML
- CSS
- PHP
- MVC
- HTTP statuses
- DOM
- JavaScript
- jQuery
- Ajax
- security
- AI
- ...

Quiz 1

- http://cdn.cs50.net/2015/fall/quizzes/1/yale.html

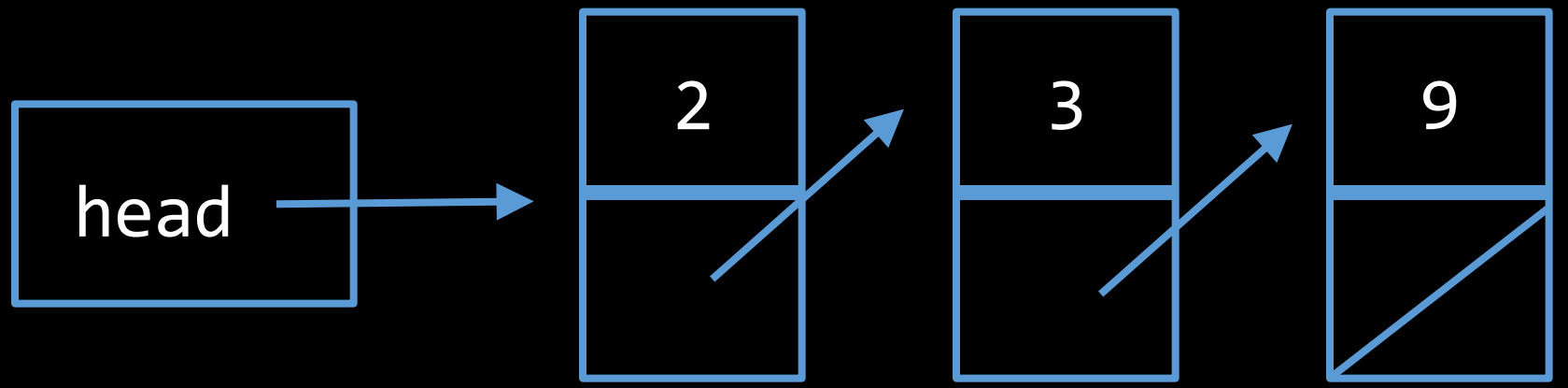
- Wednesday
  - If your LAST name starts with A – N, go to the law school auditorium
  - If your LAST name starts with O – Z, go to SSS 114
- Thursday
  - Go to SSS 114

Quiz 1

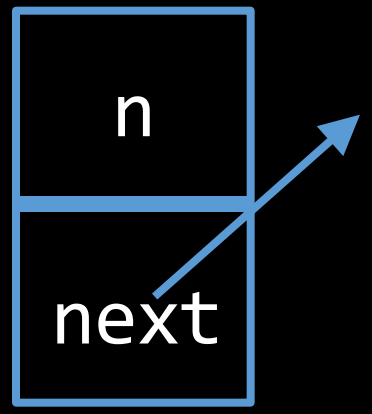
- Tips
  - Practice writing code by hand
  - Practice HTML/Javascript in the Google Chrome developer tools
  - Throw everything you have to memorize onto your cheat sheet

  Quiz 1 historically has been more difficult than Quiz 0…

Linked Lists

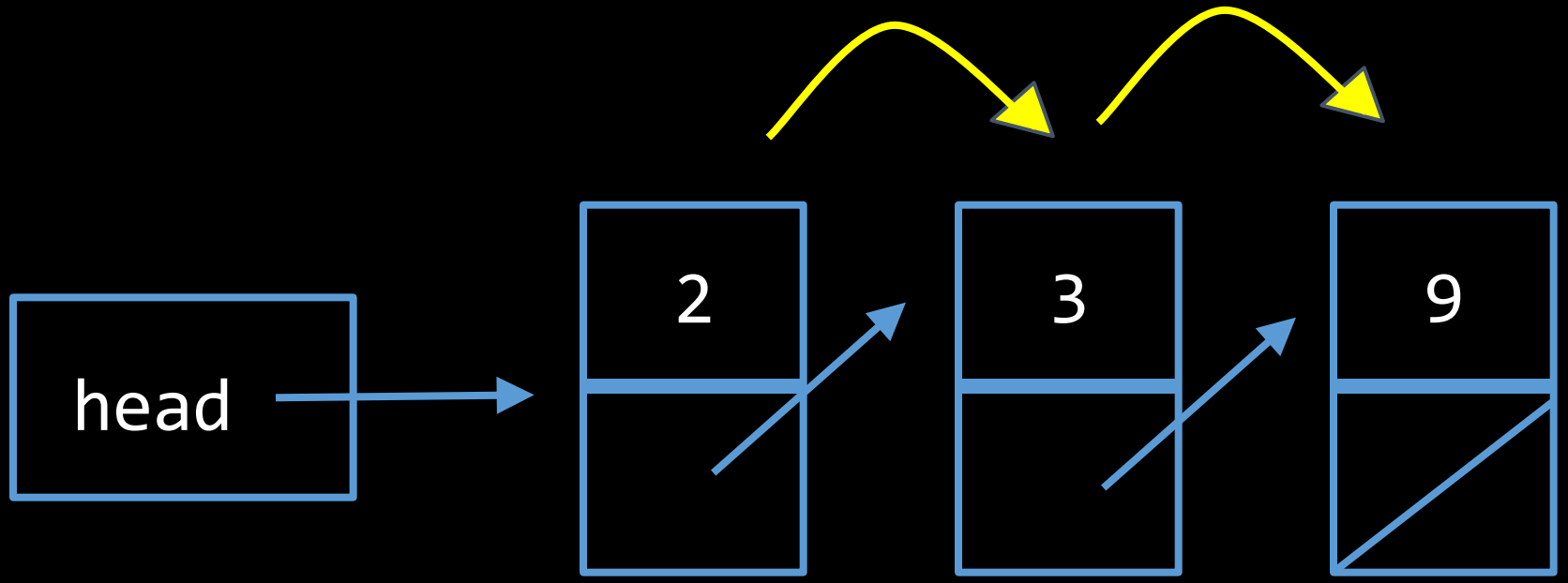


Nodes



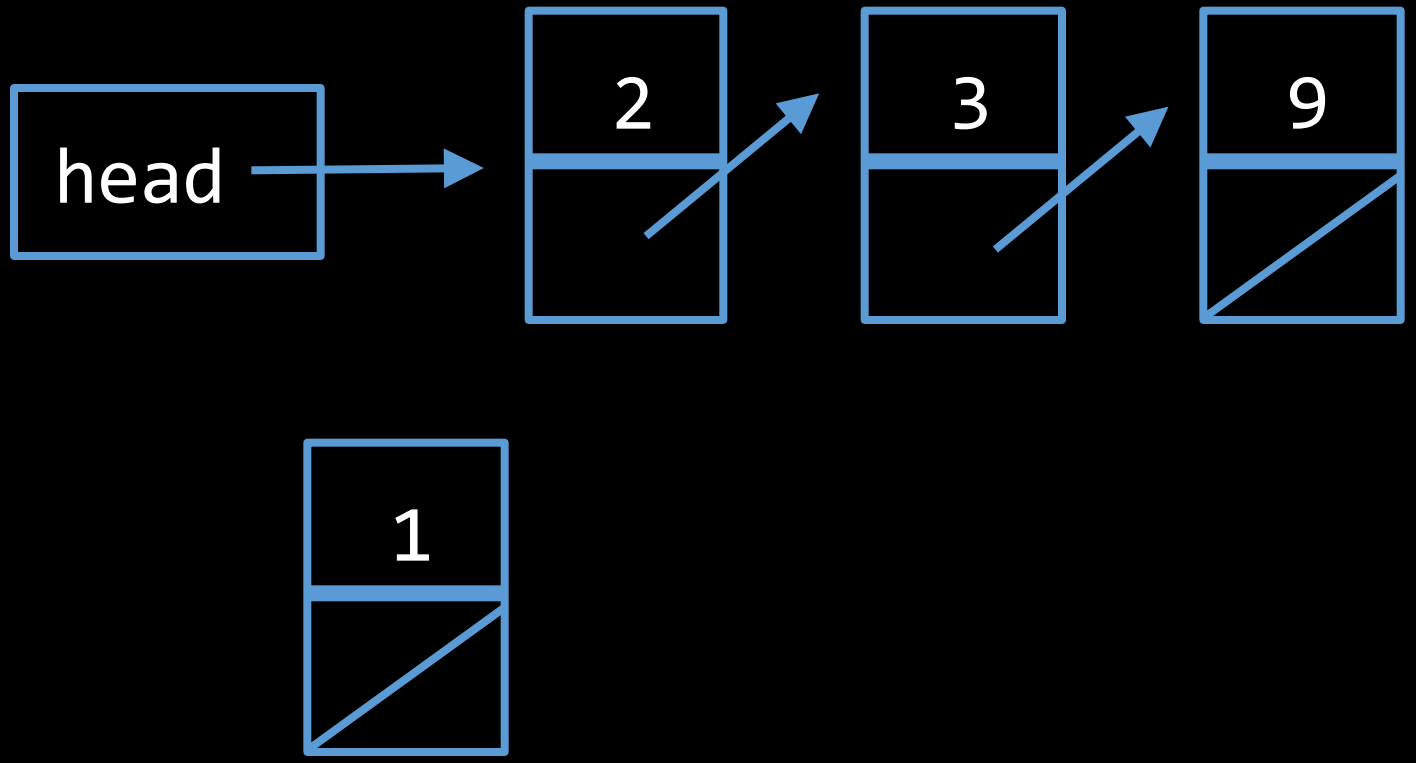

```
typedef struct node
{
        int n;
        struct node* next;
}
node;
```

Search

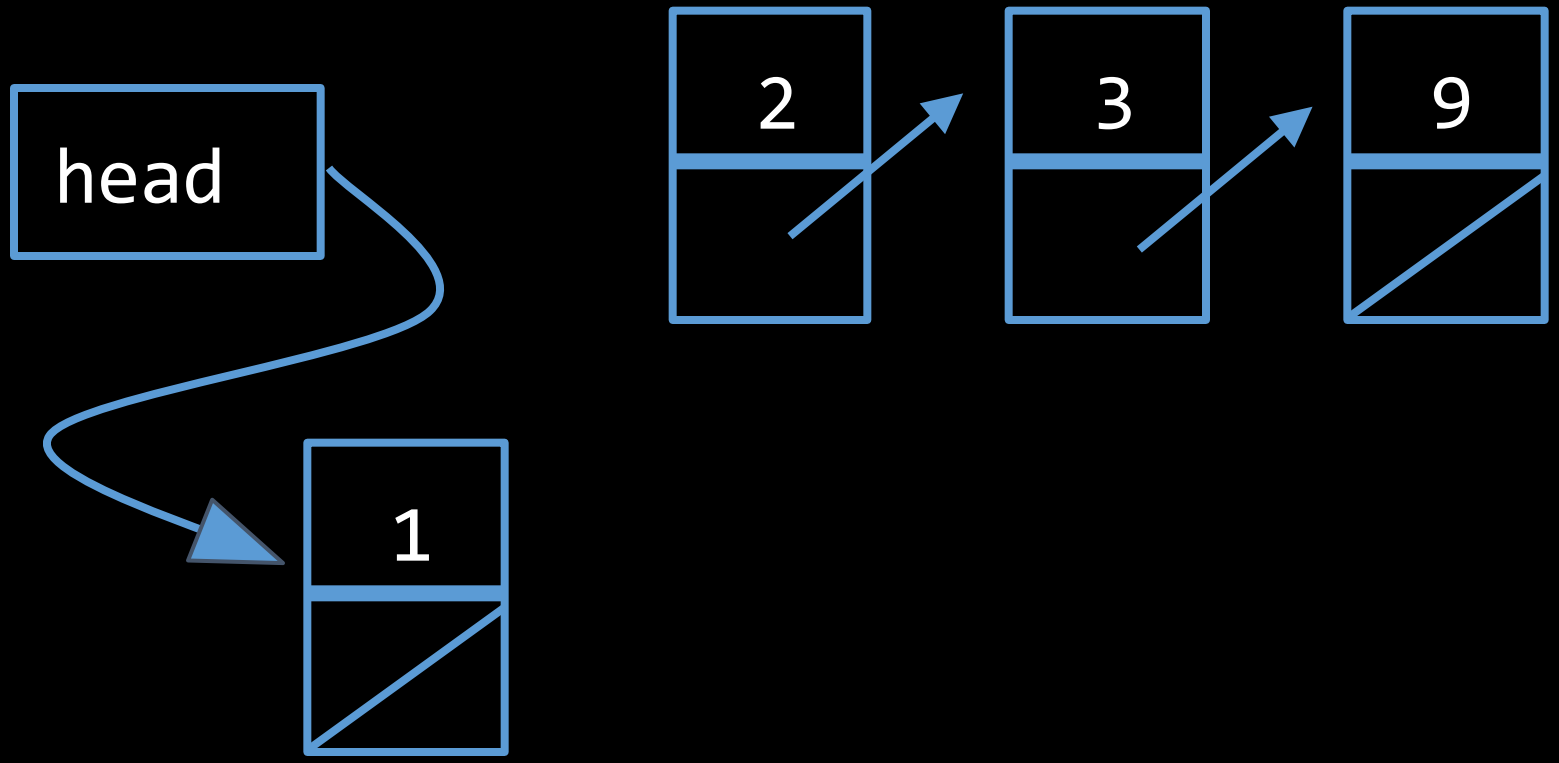


```
bool search(node* list, int n)
{
    node* ptr = list;

    while (ptr != NULL)
    {
            if (ptr->n == n)
      {
                return true;
      }
            ptr = ptr->next;
    }
    return false;
}
```

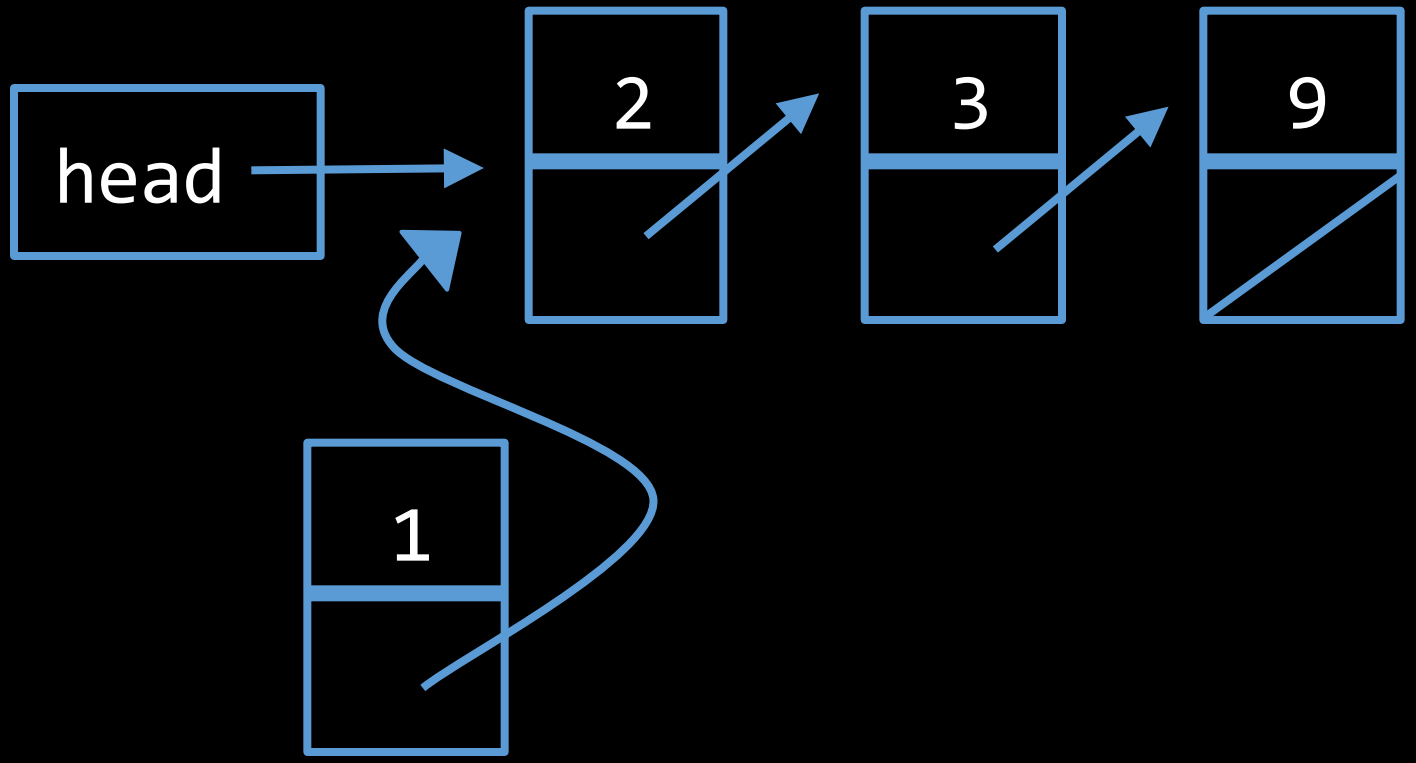
Insertion



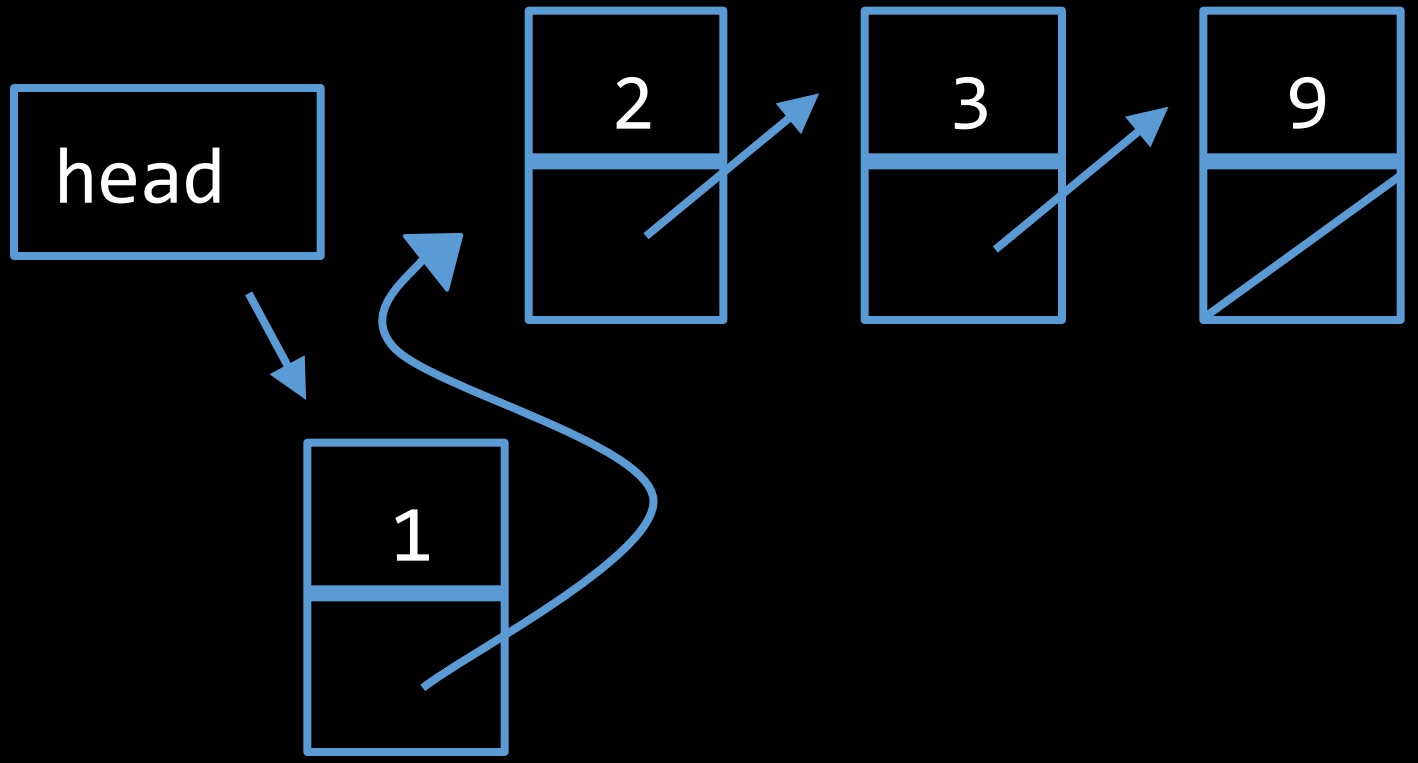
Insertion



Insertion



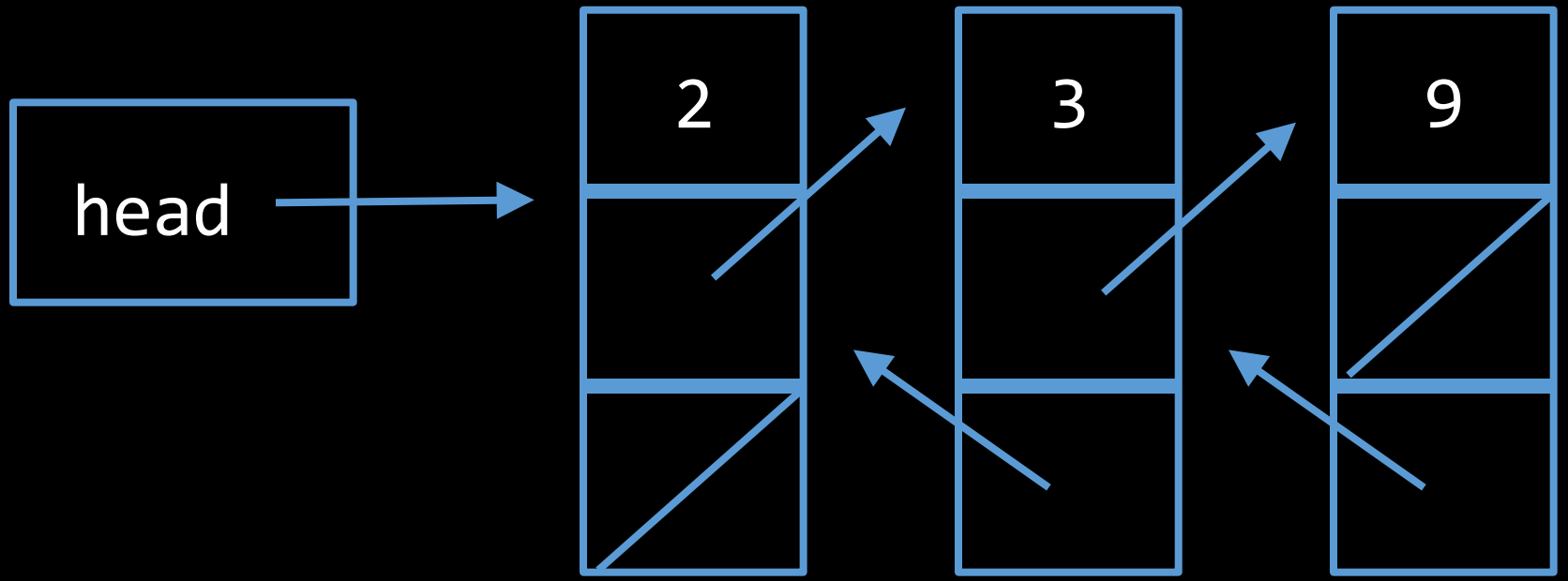
Insertion



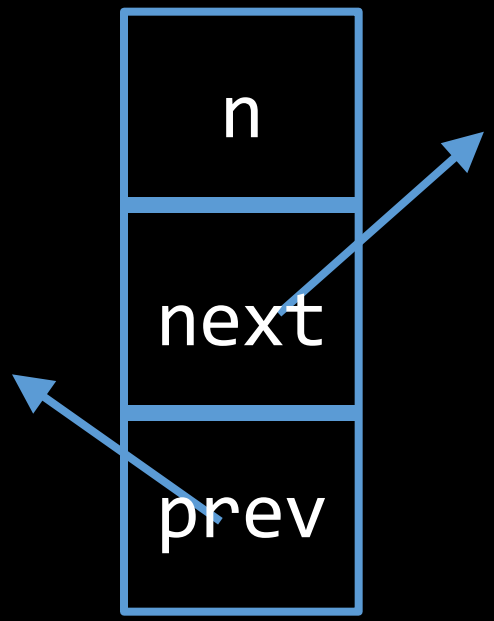
```c
void insert(int n)
{
    // create new node
    node* new = malloc(sizeof(node));

    // check for NULL
    if (new == NULL)
    {
        exit(1);
    }
    // initialize new node
    new->n = n;
    new->next = NULL;

    // insert new node at head
    new->next = head;
    head = new;
}
```

Inserting into the middle or the end?

Doubly Linked Lists



DLL Nodes

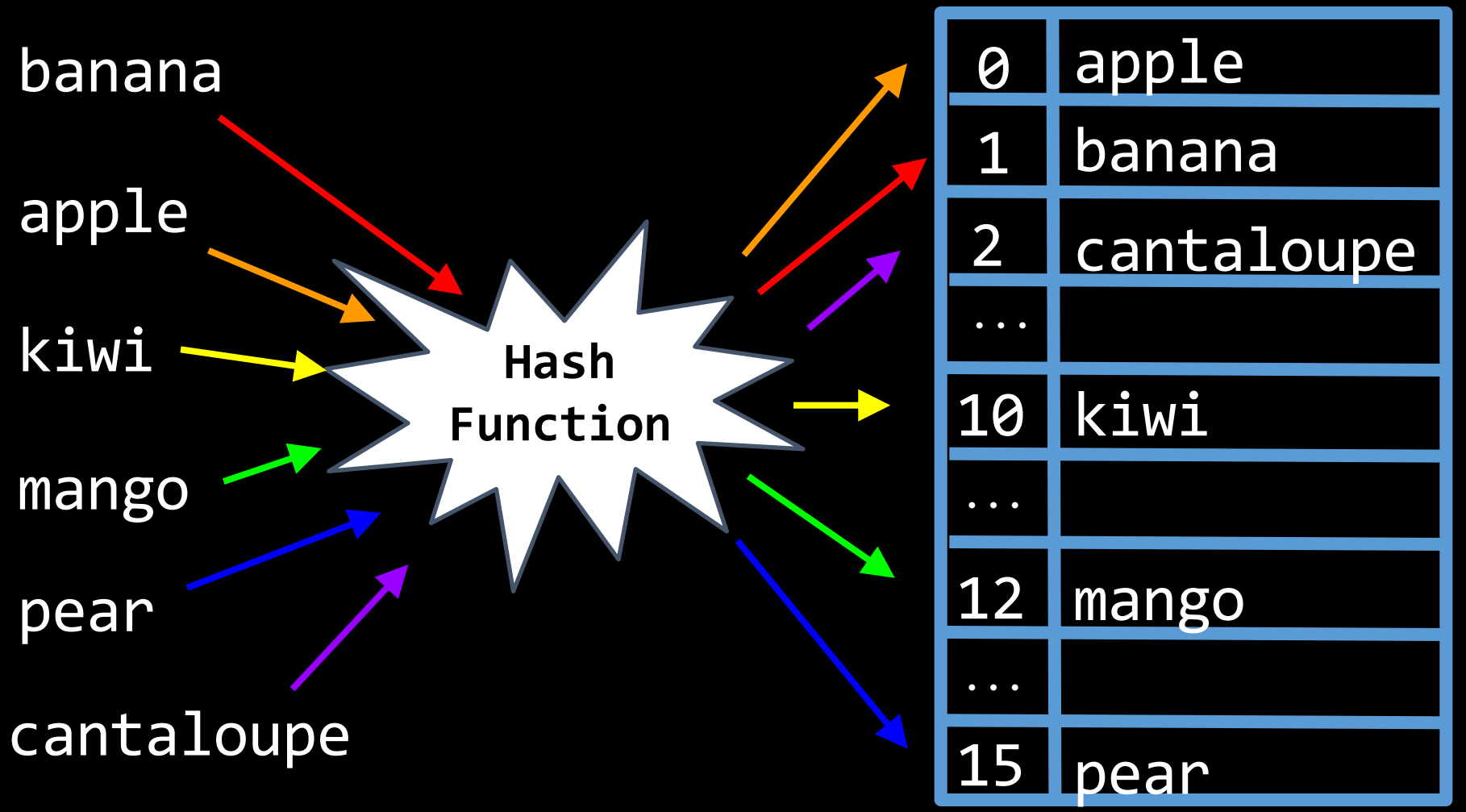



```
typedef struct  node
{
        int n;
        struct node* next;
        struct node* prev;
}
node;
```

Linked Lists

- benefits of linked lists
  - unlike arrays, size changes dynamically
  - useful for hash tables
- basic operations
  - all $\Omega(1)$
  - insert $O(1)$, delete $O(n)$, search $O(n)$
    - assuming not sorted

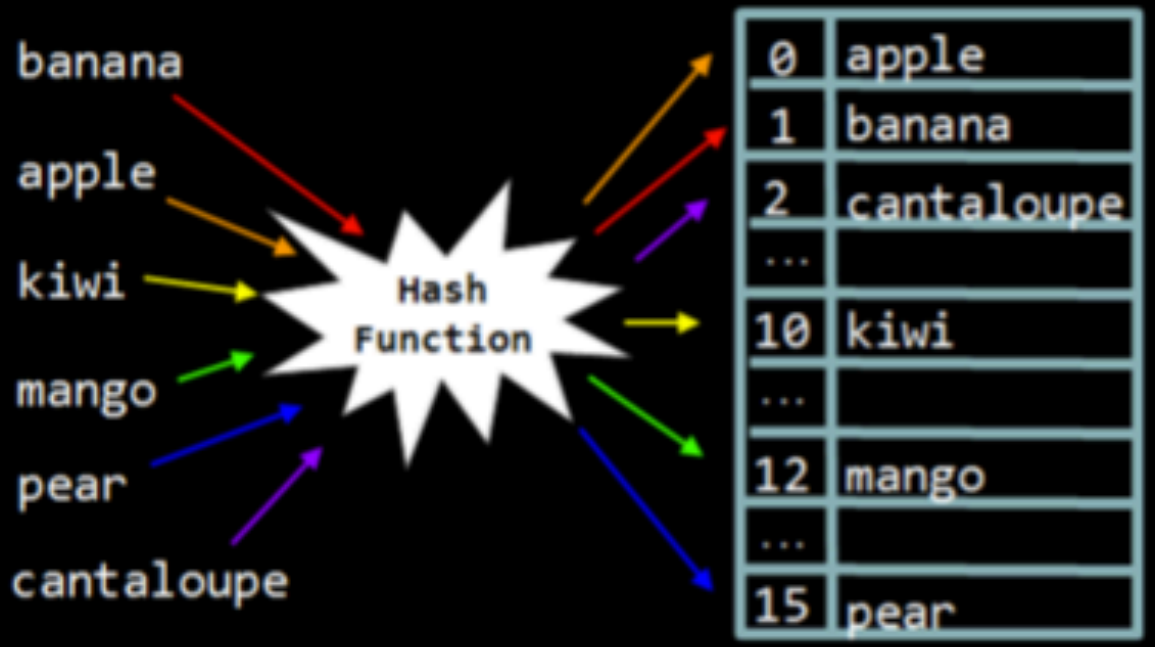
Hash Tables


banana
apple
kiwi
mango
pear
cantaloupe
Hash Function
0 apple
1 banana
2 cantaloupe
...
10 kiwi
...
12 mango
...
15 pear

Hash Function

determines where to insert or lookup a word

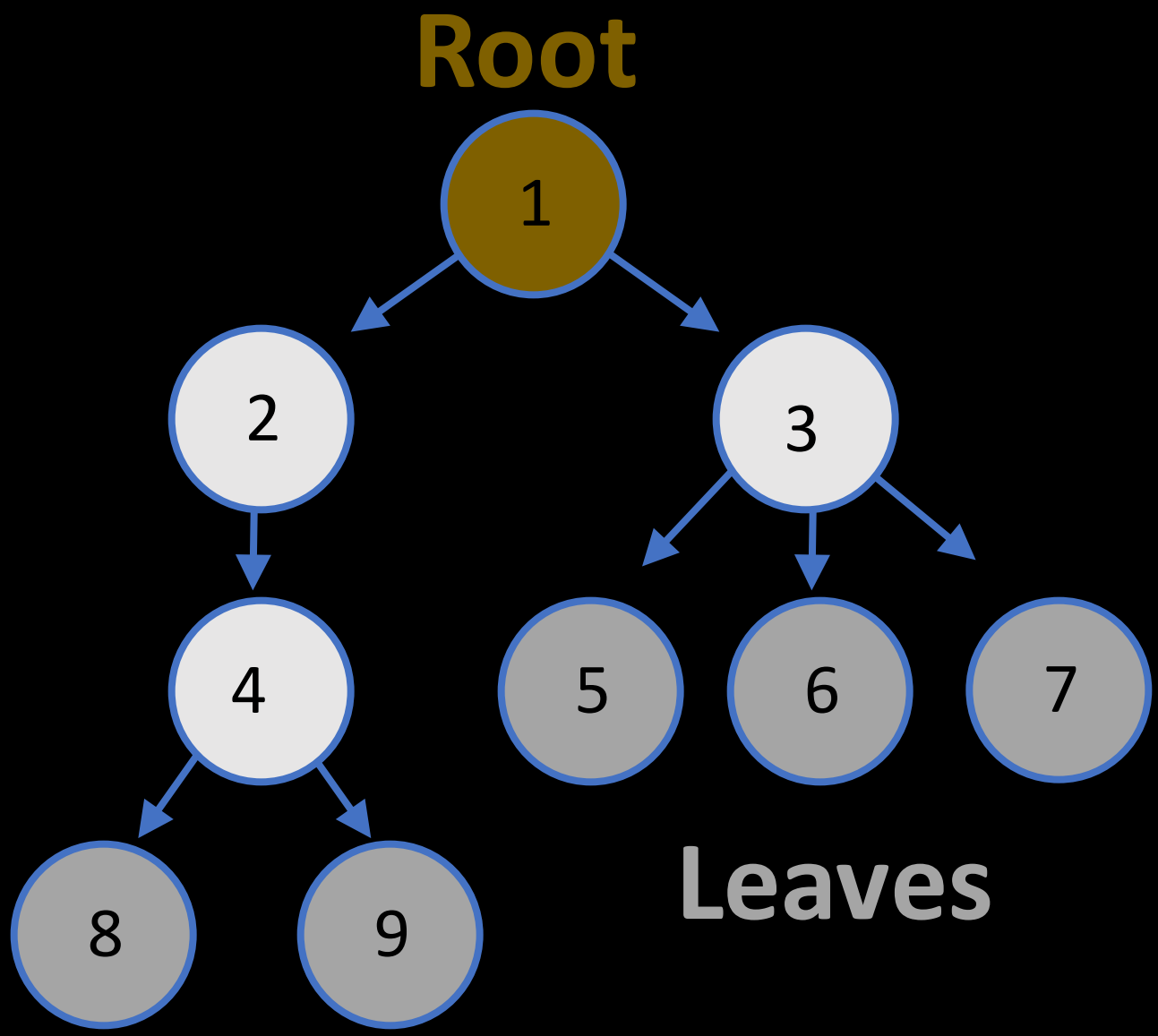
```
int hash_function(char* key)
{
    // hash on first letter of string
    int value = toupper(key[0]) - 'A';
    return value % SIZE;
}
```

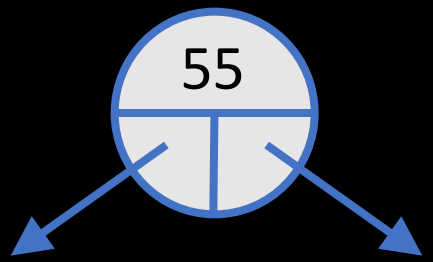


Trees

tree: a data structure in which data is organized hierarchically

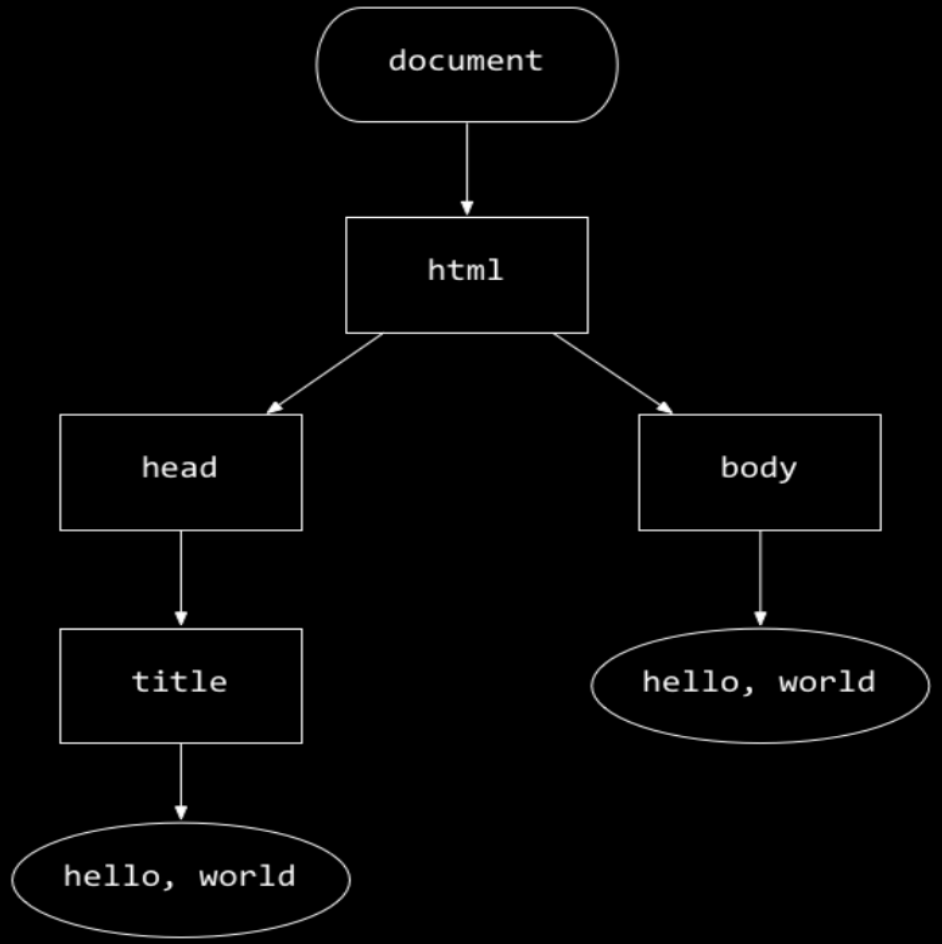
trie: special kind of tree that behaves like a multi-level hash table


Root
1
2
3
4
5
6
7
8
9
Leaves




```
typedef struct node
{
        int n;
        struct node* left;
        struct node* right;
}
node;
```

DOM: Document-Object Model




```
<!DOCTYPE html>

<html>
        <head>
                <title>hello, world</title>
        </head>
        <body>
                hello, world
        </body>
</html>
```

DOM: Document-Object Model (2)

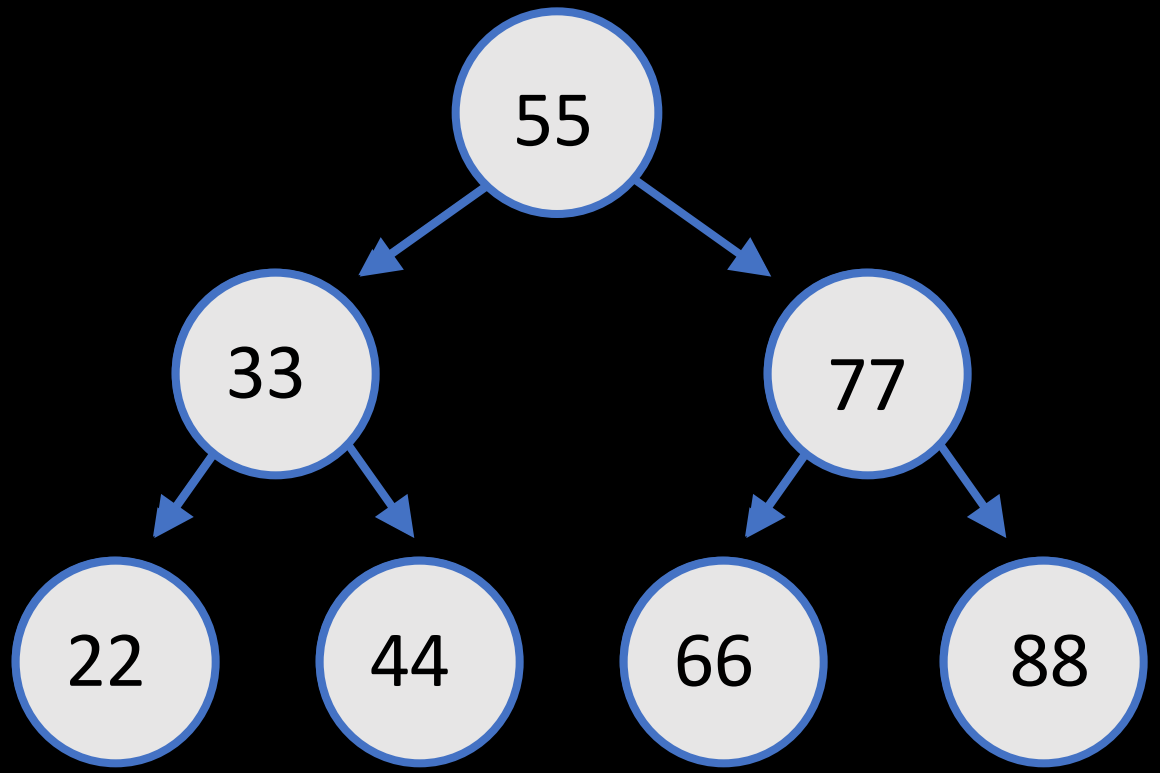
Examples

- `document.title`
- `document.body`
- `document.body.innerHTML`

Useful Functions

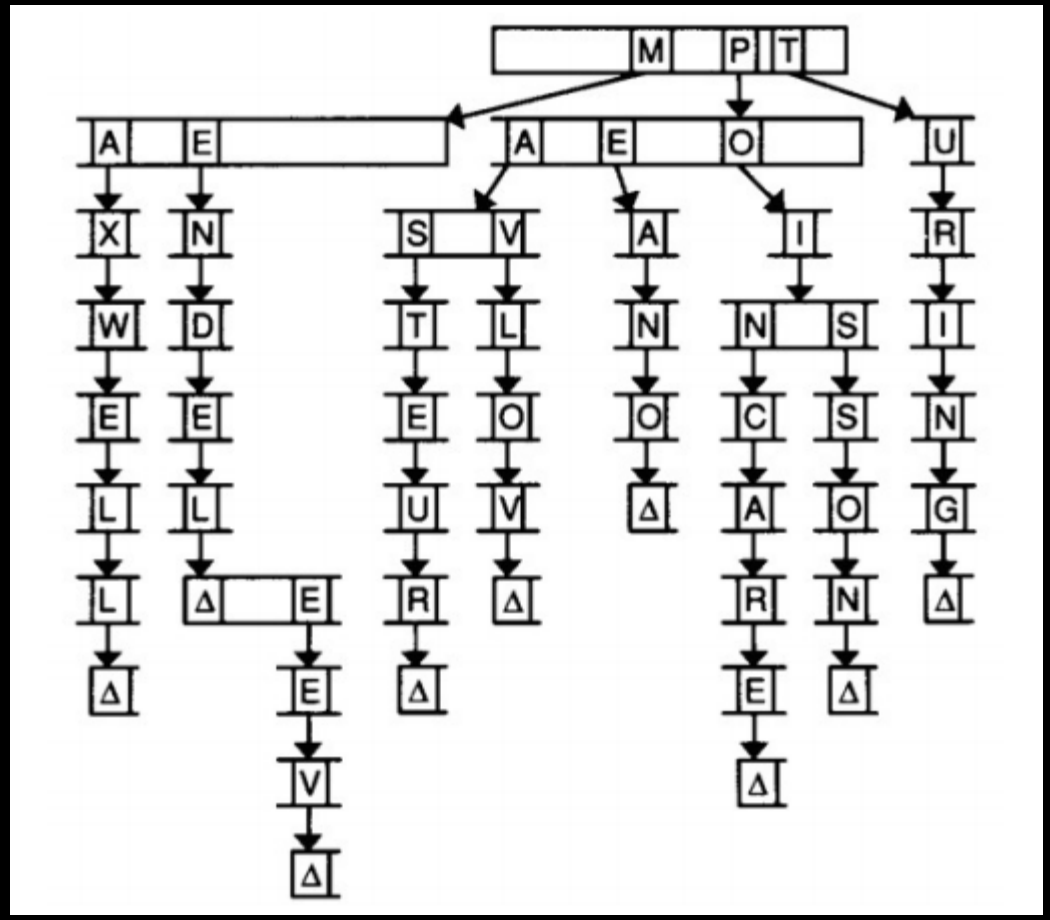
- `document.getElementById(`*`string`*`)`
- `document.getElementsByClassName(`*`string`*`)`
- `document.getElementsByTagName(`*`string`*`)`

Binary Search Tree

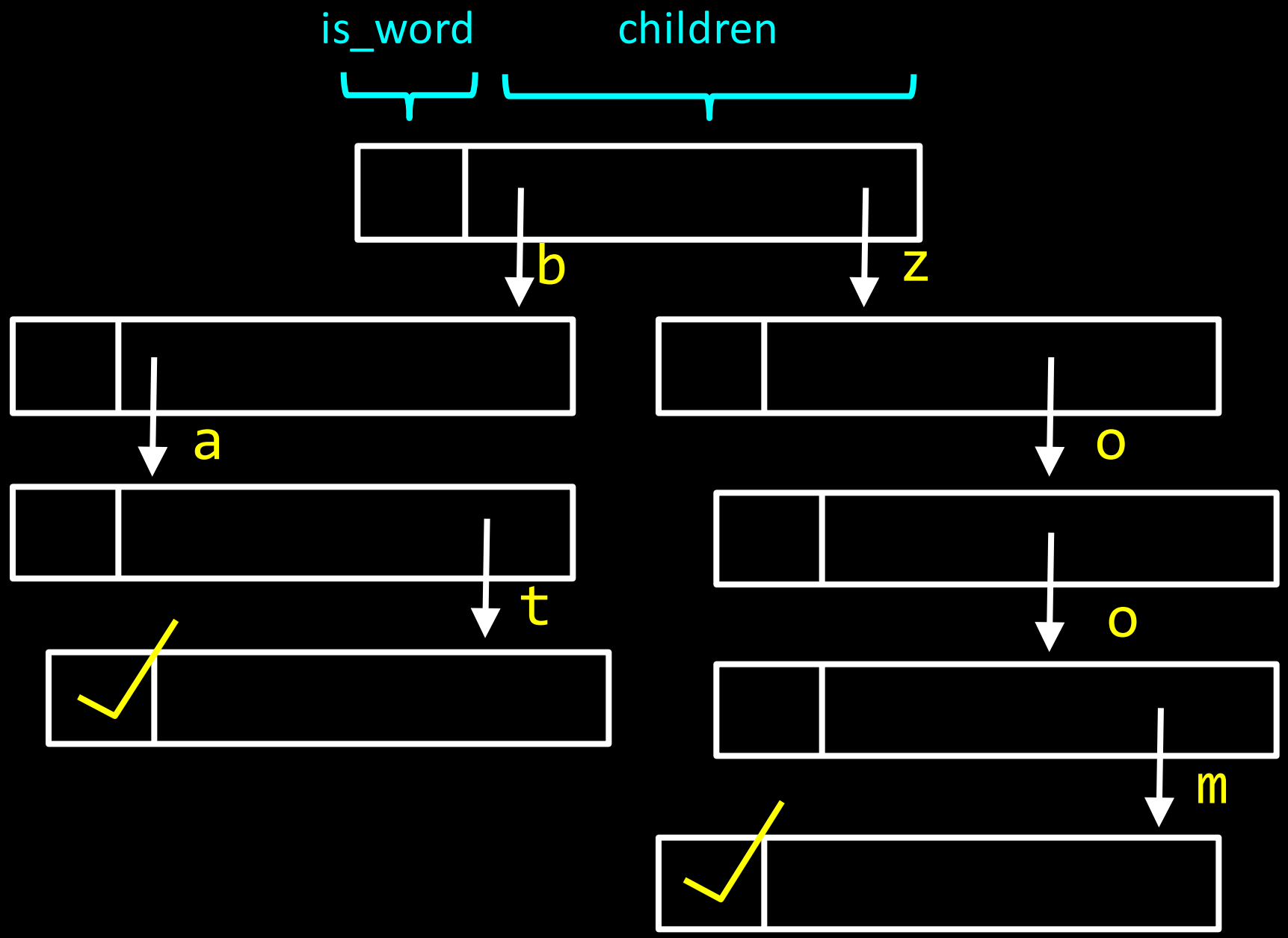


```
bool search(node* root, int val)
{
    if root is NULL
        return false.

    if root->n is val
        return true.

    if val is less than root->n
        search left child

    if val is greater than root->n
        search right child
}
```

Tries

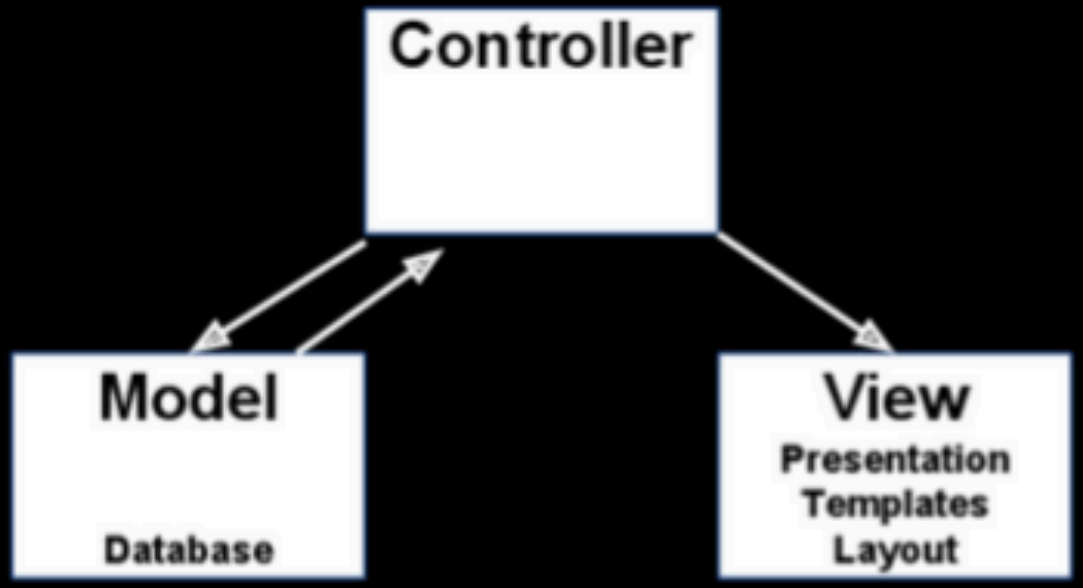

M P T
A E
A E O
U
X N
S V
A
I
R
W D
T L
N
N S
I
E E
E O
O
C S
N
L L
U V
Δ
A O
G
L Δ E
R Δ
R N Δ
Δ E
Δ
E Δ
V
Δ

```
typedef struct node
{
    // marker for end of word
     bool is_word;

     // pointers to other nodes
     struct node* children[27];
}
node;
```


is_word
children
b
z
a
o
t
o
m

MVC

- design paradigm
- way of organizing and thinking about code

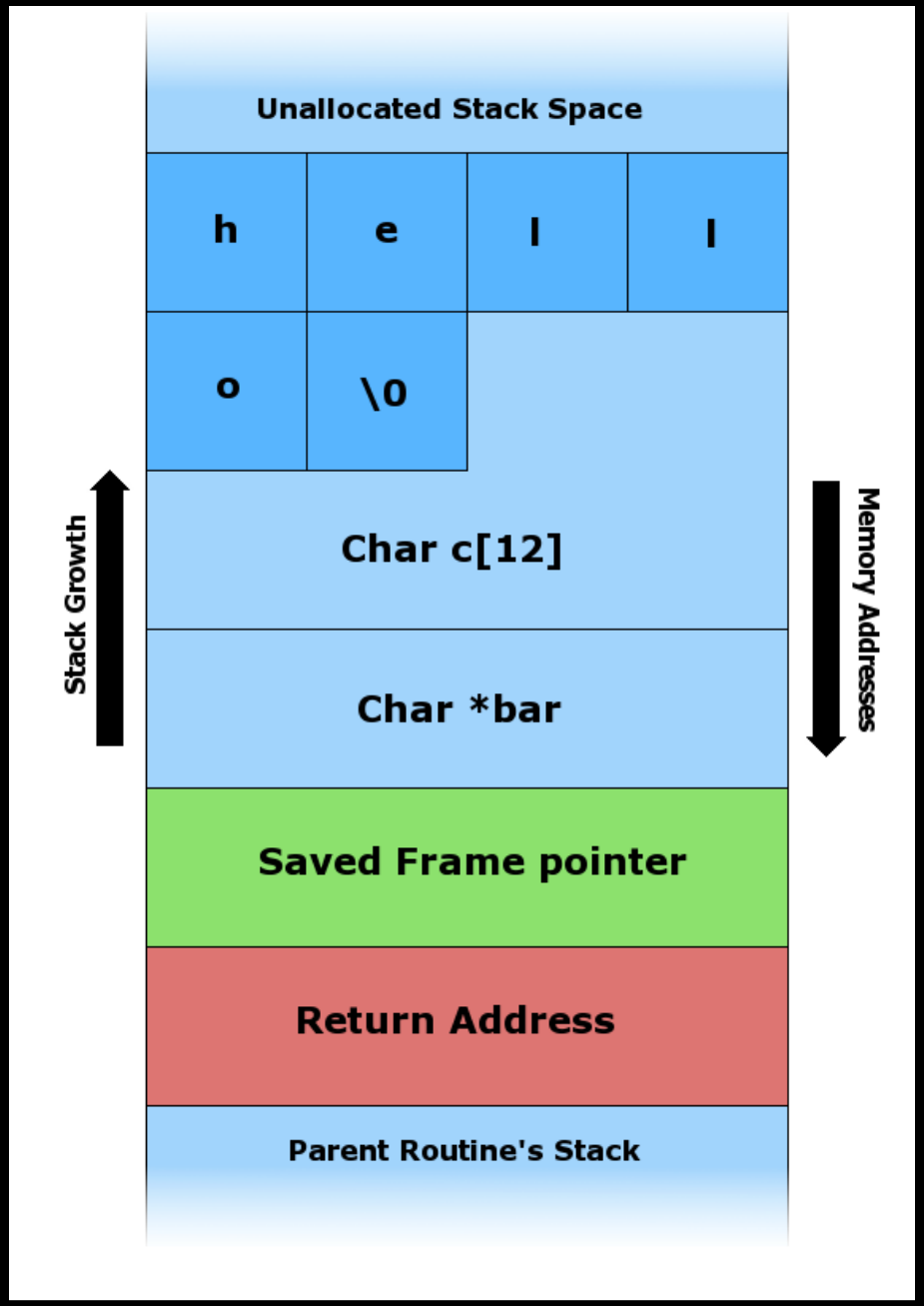


MVC

COMPONENT	FUNCTION	EXAMPLE
Model	- Persistent storage of information - Managing and organizing data	- MySQL database - Data files
View	- Presentation of information to user - User interface	- HTML - Minimal PHP (e.g., for iterating over data to print it out)
Controller	- Handles user requests, gets information from the model	- PHP

Buffer Overflow Attack

```
#include <string.h>

void foo(char* bar)
{
   char c[12];
   memcpy(c, bar, strlen(bar));
}

int main(int argc, char* argv[])
{
   foo(argv[1]);
}
```



Buffer Overflow Attack

```
void foo(char* bar)
{
   char c[12];
   if (bar != NULL)
   {
      int n = strlen(bar);
      if (n <= 12)
      {
         memcpy(c, bar, n);
      }
   }
}
```

Other Types of Attack

- Session hijacking
- SQL Injection Attack
- Manipulating header data

Questions?

Andi's Pump Up Speech Part II