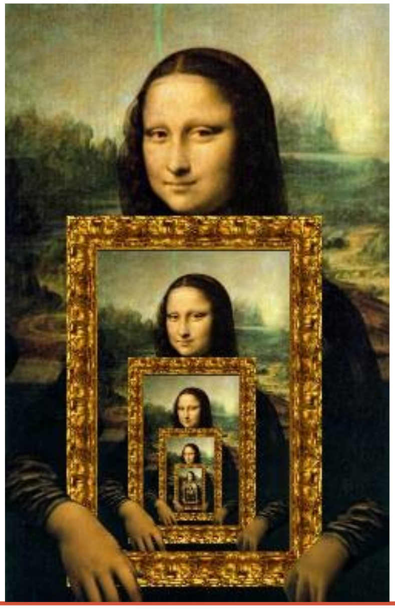
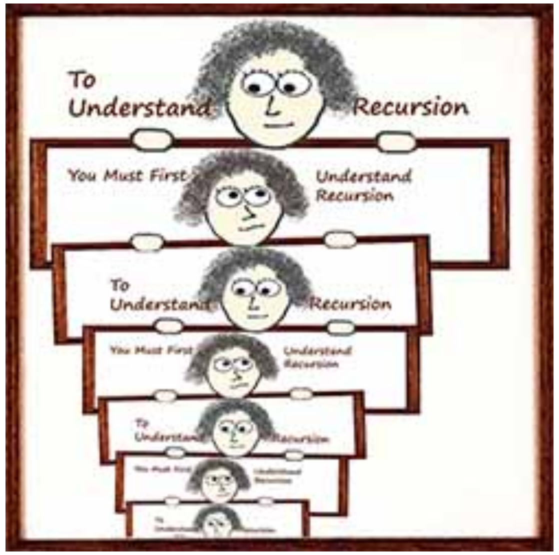
RECURSION

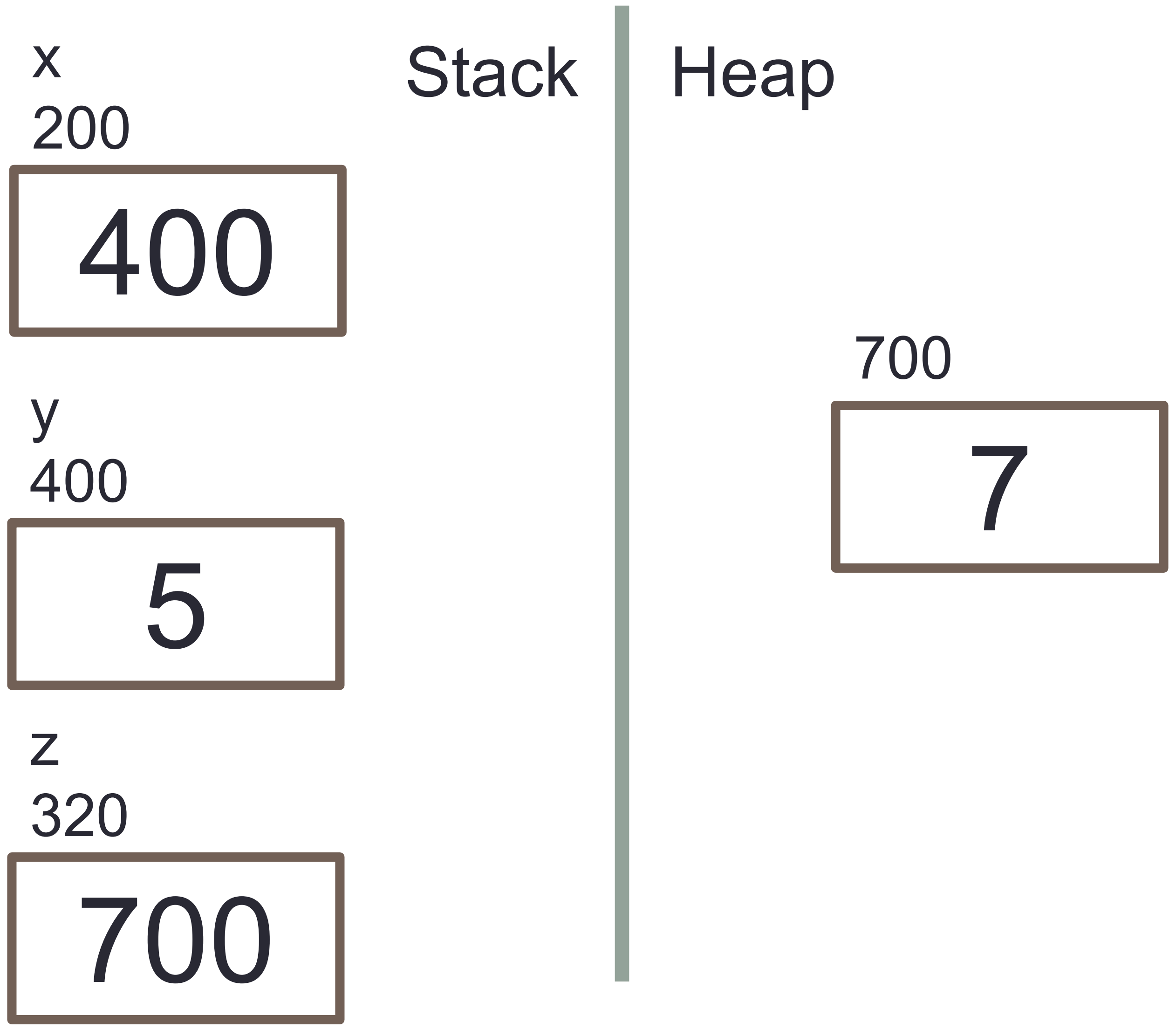


Problem Solving with Computers-I


To Understand Recursion
You Must First Understand Recursion
To Understand Recursion
You Must First Understand Recursion
To Understand Recursion

Stack & Heap Example

```
int *x;
int y = 5;
x = &y;

int *z = new int;
*z = 7;
```



Midterm 2 Question 7a

```
T findBestElement(T arr[], int size) {
  // …

  T guess = arr[0];
  for (int i = 1; i < size; i++) {
    if (betterThan(arr[i], guess))
      guess = arr[i];
  }

  return guess;
}
```

- T could be any type (int, bool, TideLevel, etc.)

```
bool betterThan(T a, T b);
```

Thinking recursively!

- Many structures in nature and CS that are recursive
- A recursive solution to a problem is all about describing the problem in terms of a ***smaller*** version of itself!

Keys to recursion:

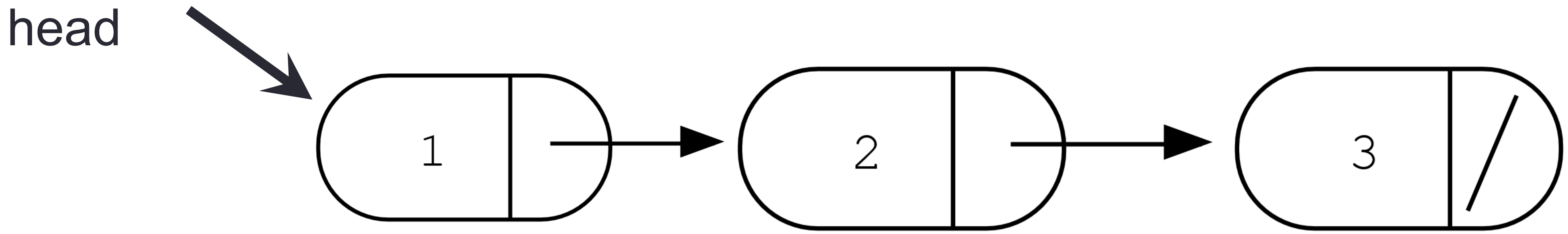
1. the problem must get smaller

2. the problem can’t get smaller forever

Thinking recursively!

1. Base case: solve the small*est* version(s) of the problem
2. Recursive case: describe the problem in terms of itself!
    - Assume you have a solution for a smaller input size!
    - Describe the problem in terms of a smaller version of itself.

Example problem: Print all the elements of a linked-list backwards!



```
What is the smallest version of this problem?
```

Step 1: Base case!

```
//Write code for the smallest version of the problem
void printBackwards(Node * head) {








}
```

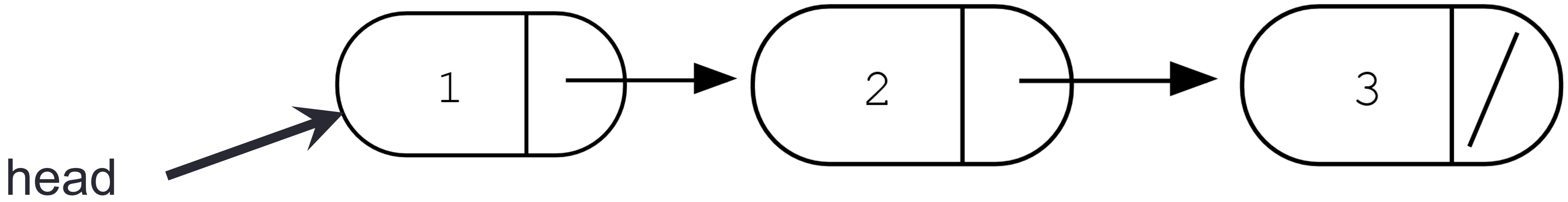
Step 2: Write the recursive case!

- Assume you have a solution for a smaller version of the problem!!!!
- Describe the problem in terms of a smaller version of itself

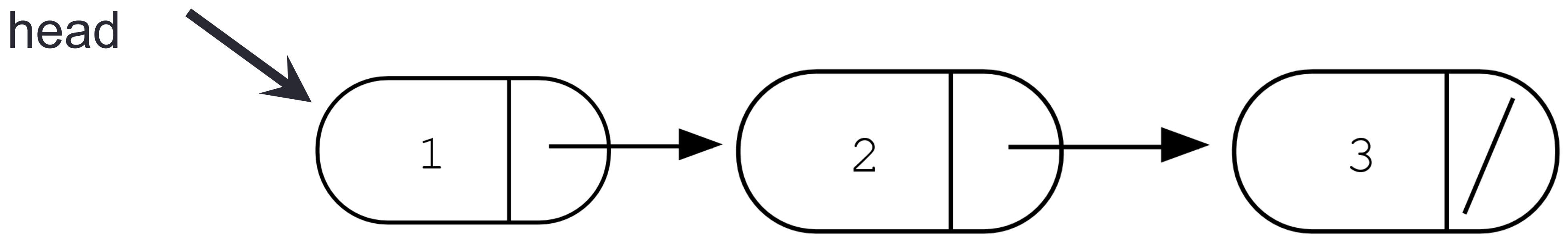
```
void printBackwards(Node * head){
       if (head == NULL)  //Base case
              return;



}
```

Q: What is the right order?

(A) Print the head's data, then make the recursive call
(B) Make the recursive call, then print the head's data



Example 2: Find the sum of the elements of a linked-list



Step 1: Base case!

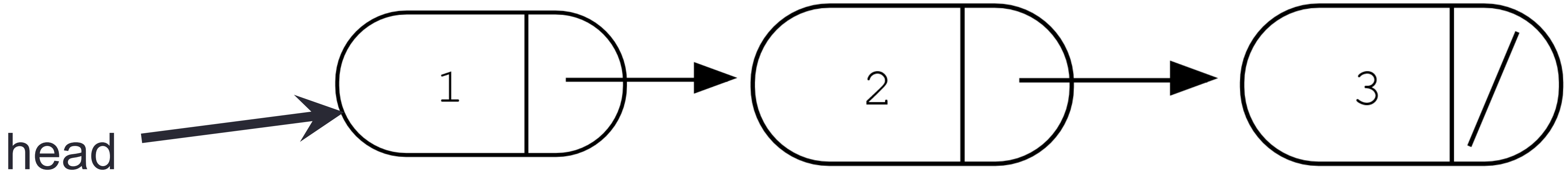
- Write code for the smallest version of the problem

```
int sum(Node * head) {








}
```

Step 2: Write the recursive case !

- Assume you have a solution for a smaller version of the problem!!!!
- Describe the problem in terms of a smaller version of itself

```
void sum(Node * head){
    if (head == NULL)  //Base case


}
```



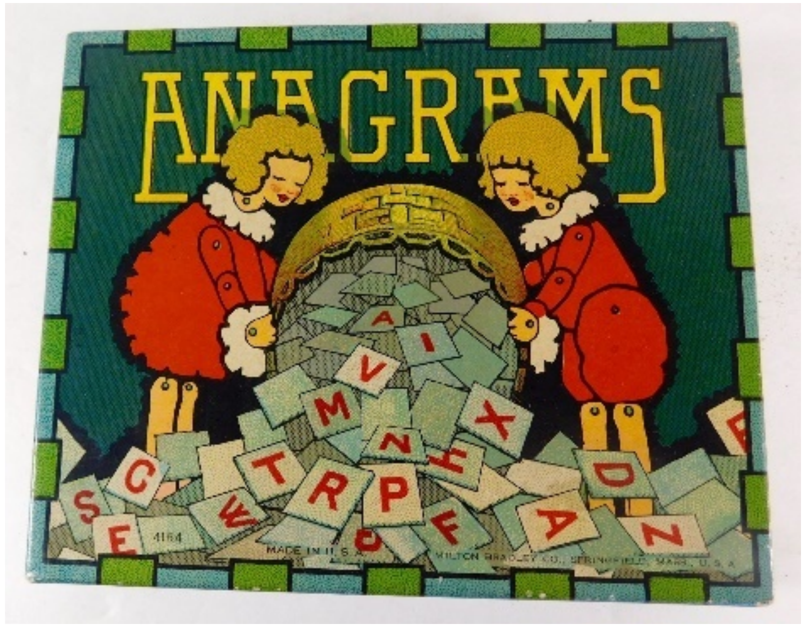
Example 3: Backwards with arrays

name	‘B’	‘o’	‘n’	‘d’	‘0’	‘0’	‘7’

```
void printElementsBackwards(char *arr, int len){

    if(len<=0){ //Base case
         return;
    }
   //Write your code here



}
```

Anagrams and Palindromes

```
bool isAnagram(string s1, string s2)
```



Diba == Adib
Rats and Mice == In cat's dream
Waitress == A stew, Sir?

```
bool isPalindrome(const string s1) //recursive
bool isPalindrome(const char *s1) //recursive
bool isPalindromeIterative(const char *s1) //iterative
```

deTartraTED
WasItACarOrACatISaw

Why don't we pass the length of the string?