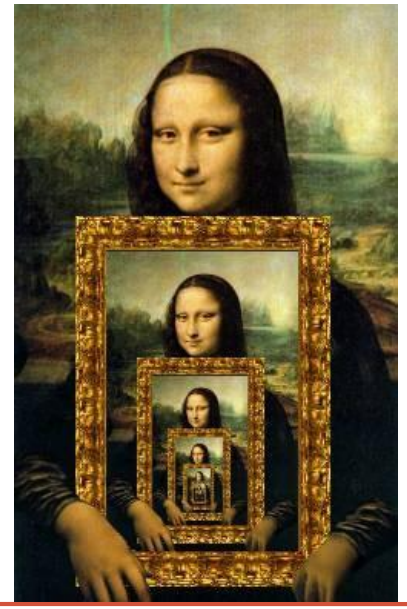
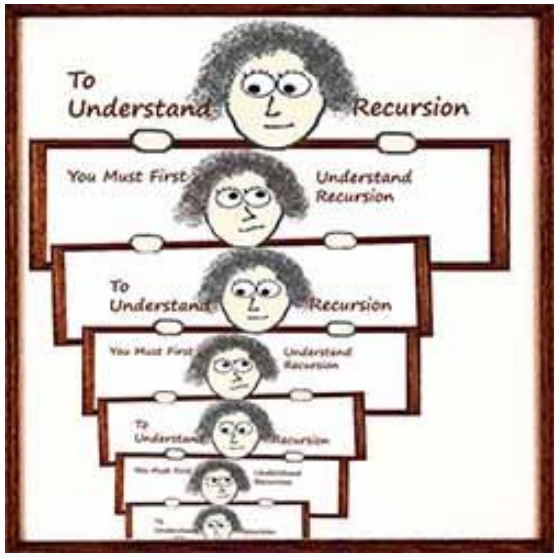
RECURSION



Problem Solving with Computers-I



https://ucsb-cs16-sp17.github.io/

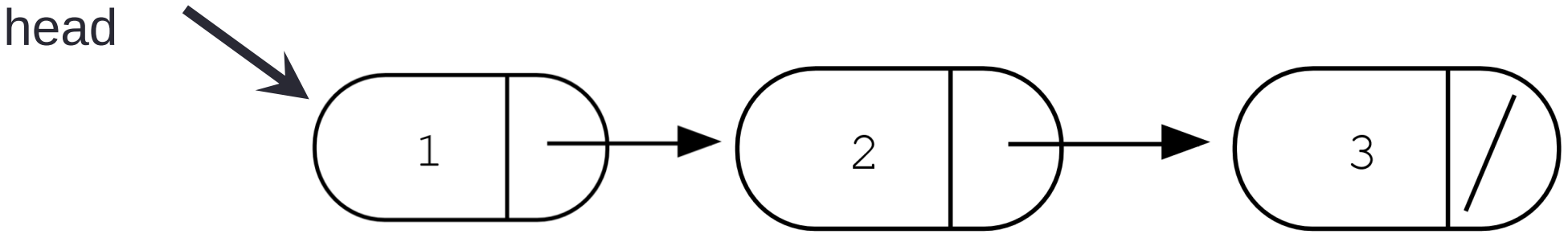
Thinking recursively!

- 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!

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?

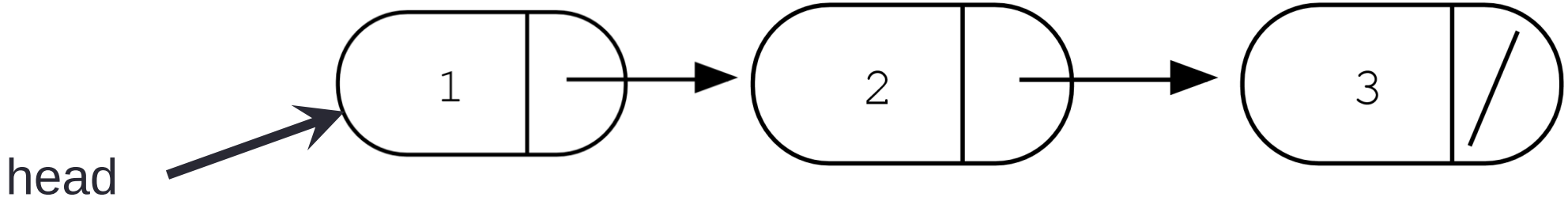
Step 1: Base case!

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




}
```

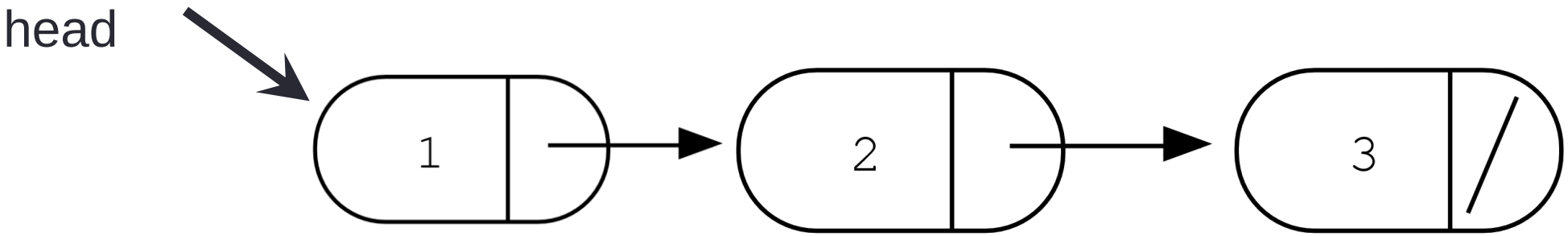
Step 2: Write the recursive case !

- 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;
```



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



Step 1: Base case!

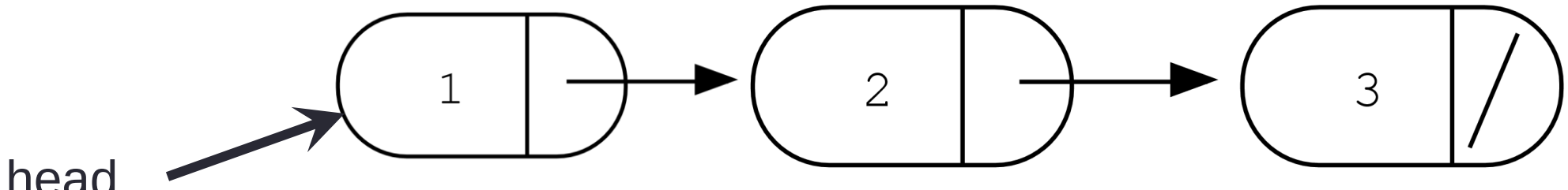
- Write code for the smallest version of the problem

```
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


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

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



}
```

C-Strings

Q1: How are ordinary arrays of characters and C-strings similar and how are they dissimilar?

Which of the following is not a C string?

A. `char mystr[5] = “John”;`

B. `char mystr[] = “Mary”;`

C. `const char *mystr = “Jill”;`

D. `char mystr[4]= {‘J’, ‘i’, ‘l’, ‘l’};`

Q2: Which of the following statements about the given code is FALSE?

```
char s1[5] = "Mark", s2[5] = "Jill";
for (int i = 0; i <= 5; i++)
    s1[i] = s2[i];
if (s1 != s2) s1 = "Art";
```

A. There is an out of bound access in the for loop
B. The entire for loop can be replaced by s1 = s2;
C. In the if statement, the logic for comparing two strings is incorrect.
D. The body of the if statement is incorrect: cannot change the base address of an array

C String Standard Functions #include <cstring>

```
char s1[5] = "Mark", s2[5] = "Jill";
for (int i = 0; i <= 5; i++)
    s1[i] = s2[i];
if (s1 != s2) s1 = "Art";
```

- **`int strlen(char *string);`**
  - Returns the length not counting of `string` the null terminator
- **`int strcmp(char *str1, char *str2);`**
  - return 0 if `str1` and `str2` are identical (how is this different from `str1 == str2`?)
- **`int strcpy(char *dst, char *src);`**
  - copy the contents of string `src` to the memory at `dst`. The caller must ensure that `dst` has enough memory to hold the data to be copied.
- **`char* strcat(char *s1, char *s2);`**
  - concatenate the contents of string `s2` to `s2`and returns pointer to resulting string

Q3: What is the output of the following code? (solo vote)

```
char s1[4] = "abc", s2[4] = “EFG";

if (strcmp(s1, s2)) cout << "Hi!";

else cout << "Hey!";
```

A. Hi!
B. Hey!
C. Compiler error
D. Runtime error

C strings vs. String class: What is the output of the code?

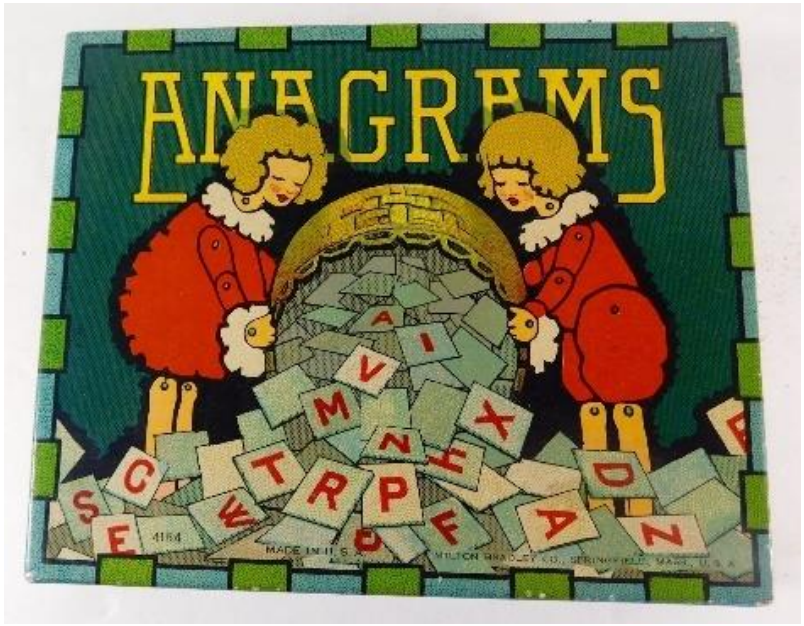
```
string s1 = "Mark";
string s2 = "Jill";
for (int i = 0; i <= s1.length(); i++)
     s2[i] = s1[i];
if (s1 == s2) s1 = "Art";
cout<<s1<<“ ”<<s2<<endl;
```

A. Mark Jill
B. Mark Mark
C. Art Mark
D. Compiler error
E. Run-time error

The C++ string class methods

```
string fruit = ”Apple";
int len = fruit.length();
int pos= fruit.find(‘l’);
string part= fruit.substr(1,3);
fruit.erase(2,3);
fruit.insert(2,“ricot”);
fruit.replace(2,5,“ple”);

Check out ctype for checks and conversions on
characters
fruit[0]= tolower(fruit[0]);
isalpha(fruit[0])
```

Lab 08: 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?

Next time

- Dynamic memory pitfalls
- Advanced problems in recursion involving strings