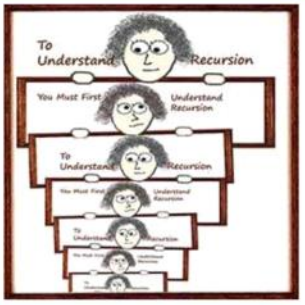
Lecture 14: Recursion

Wednesday, May 24, 2017 1:14 PM

RECURSION

Problem Solving with Computers-I

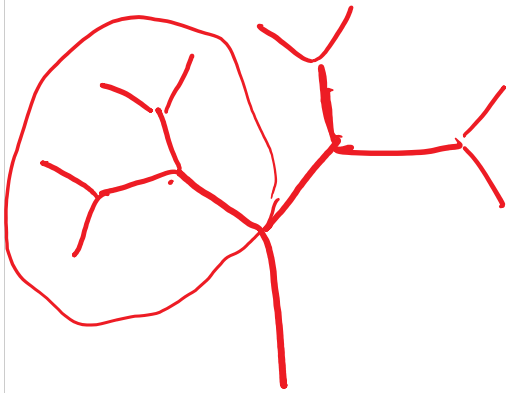
https://ucsb-cs16-sp17.github.io/

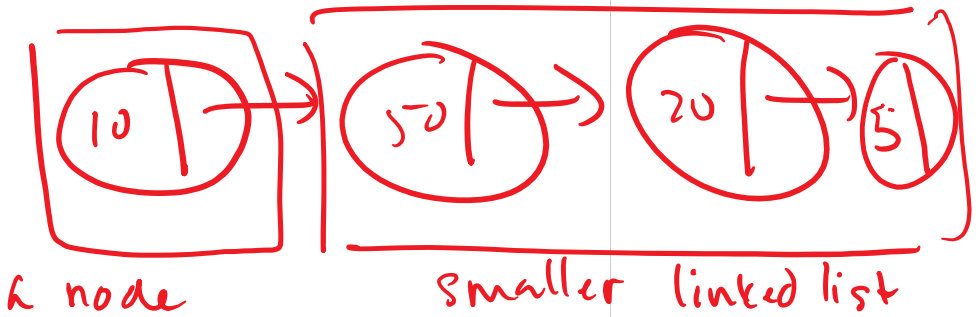


Wednesday, May 24, 2017 1:14 PM

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!

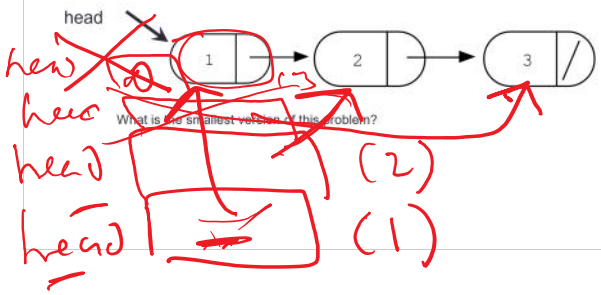




Thinking recursively!

1. Base case: solve the smallest 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!



void printBackwards(Node * head) {
if (head == 0) { // Empty list
return;

printBackwards(head->next);
cout << head->data << endl;
}

printBackwards(head)

3
2
1

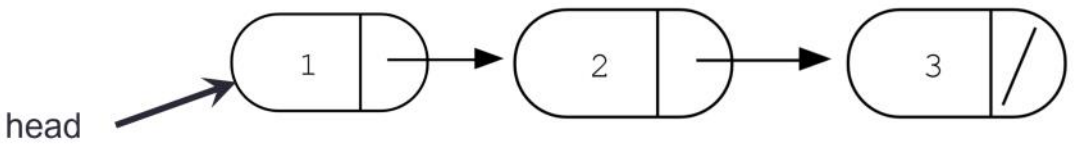
Step 1: Base case!

```
//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

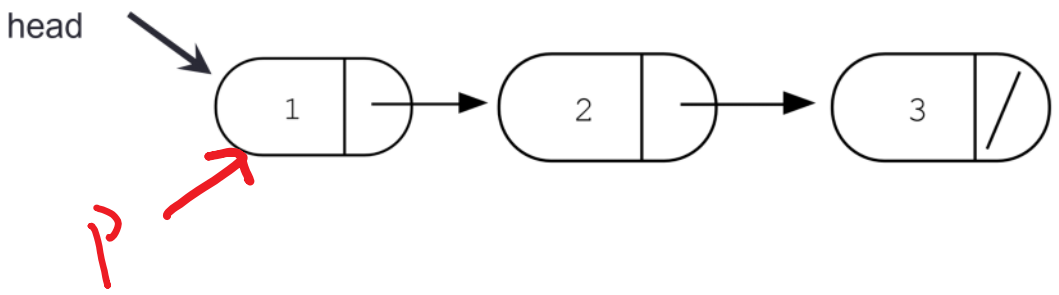
```
void printBackwards(Node * head){
      if (head == NULL)  //Base case
          return;
```



Wednesday, May 24, 2017 1:14 PM

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

```
int sum = 0;
for( Node *p = head ; p != 0 ; p = p->next ) {
    sum += p->data;
}
```



Wednesday, May 24, 2017 1:14 PM

Step 1: Base case!

- Write code for the smallest version of the problem

```
int sum(Node * head){
    if ( head == 0 ) {
        return 0;
    }
    int result = sum(head->next);
    return result + head->data;
}
```

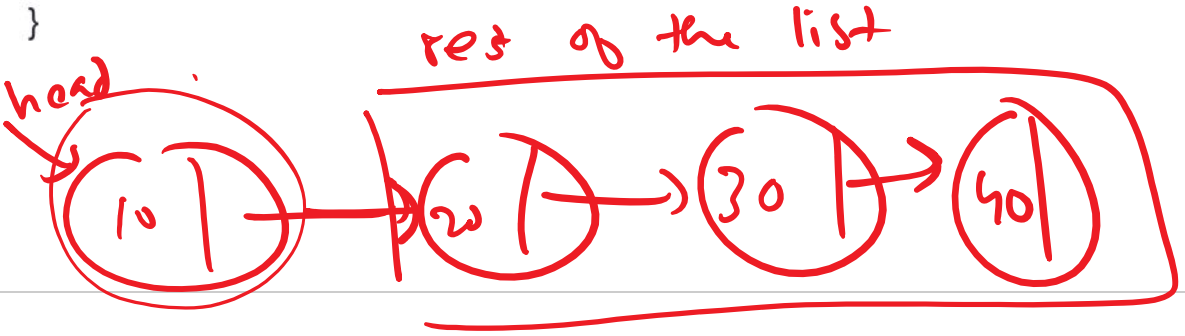
A. return 0;

B - sum = sum + head->data; ← compiles! segfault

C. return sum; ← compile error

D : return; ←

E - cout << 0; ←

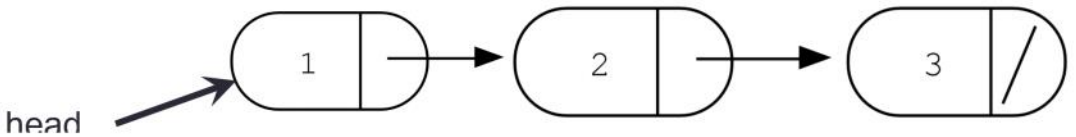


Wednesday, May 24, 2017 1:14 PM

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
```



Wednesday, May 24, 2017 1:14 PM

Example 3: Backwards with arrays

arr

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

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

}
```

printElementsBackwards(arr+1, len-1);
cout << arr[0];

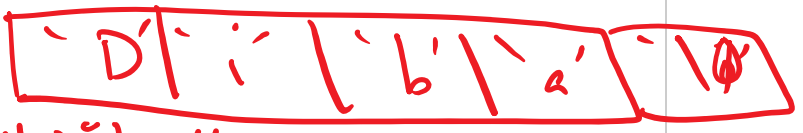
Wednesday, May 24, 2017 1:14 PM

C-Strings

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

C-String is an array of characters where the last character is '\0'

```
char name[] = {'D', 'i', 'b', 'a', '\0'};
```




```
char name[] = "Diba";   → C-string
char *name = "Jill";    → C string-literal
```

Wednesday, May 24, 2017 1:14 PM

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

int arr1[5];
int arr2[5];
arr1 = arr2;

copy s2 into s1
s1 = s2;

(!=) is not the right way to compare C-strings

s1 == s2; // assign values to C-strings

Wednesday, May 24, 2017 1:14 PM

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";
```

strlen(s1) → 4

- `int strlen(char *string);`
  - Returns the length not counting of `string` the null terminator
- `int strcmp(char *str1, char *str2);` strcmp(s1, s2)
  - return 0 if `str1` and `str2` are identical (how is this different from `str1 == str2`?)
- `int strcpy(char *dst, char *src);` strcpy(s1, s2)
  - 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

<cstring>

Wednesday, May 24, 2017 1:14 PM

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!";
```

String name = "Jill";

rchu

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

Wednesday, May 24, 2017 1:14 PM

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'};`

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

Can be replaced by

```
s2 = s1;
s2 = "Mark";
s1 = "Art";
```

Such assignments are okay if s1 & s2 are of type "string"

Wednesday, May 24, 2017 1:14 PM

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");
```

pos → 2, num characters → 5, replacement string → "ple"

Check out ctype for checks and conversions on characters

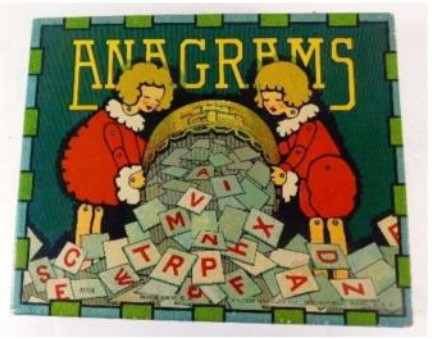
```
fruit[0]= tolower(fruit[0]);
isalpha(fruit[0])
```

converts a character to lower case

returns true if character is an alphabet (lower or upper case)

These functions will be useful for lab08

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?

Wednesday, May 24, 2017 1:14 PM

Next time

- Dynamic memory pitfalls
- Advanced problems in recursion involving strings