RECURSION

WHAT IS RECURSION?

- "A description of something that refers to itself is called a *recursive definition.*"
- In mathematics, certain recursive definitions are used all the time.
- The classic recursive example in mathematics is the definition of **factorial**.

THE FACTORIAL OF A NUMBER

- we define the factorial of a value like this:
- n! = n(n - 1)(n – 2)...(1)
- For example, we can compute
- 5! = 5(4)(3)(2)(1)
- Looking at the calculation of 5!, you will notice something interesting. If we remove the 5 from the front, what remains is a calculation of 4!. In general,
- n! = n(n - 1)!

CONTINUED…

- 4! = 4(3!) = 4(3)(2!) = 4(3)(2)(1!) =4(3)(2)(1)= 24
- You can see that the recursive definition is not circular because each application causes us to request the factorial of a smaller number. Eventually we get down to 0, which doesn't require another application of the definition. This is called a *base case for the recursion.*

CHARACTERISTICS

- All good recursive definitions have these key characteristics:
- 1. There are one or more base cases for which no recursion is required.
- 2. All chains of recursion eventually end up at one of the base cases.

CONTINUED…

- The simplest way to guarantee that these two conditions are met is to make sure that each recursion always occurs on a *smaller version of the original problem.* A very small version of the problem that can be solved without recursion then becomes the base case. This is exactly how the factorial definition works.

THE FACTORIAL FUNCTION

- If we write factorial as a function, the recursive definition translates directly into code.

```
fact(n):
    if n == 1:
        return 1 →BASE CASE! Stops calling itself
    else:
        return n * fact(n-1)
```

REVERSE STRING RECURSION

Using recursion reverse a string. A simple example would be to change “cat” into “tac”.

Remember, recursion is more easily carried out when thinking of a way to reduce the info passed back into the function.

That way, eventually we reach the base case.

How would this work in this problem?

ALGORITHM!

Pseudocode:

1. Get the last letter
2. Return that letter + the word minus that last letter.
3. When there is only 1 letter left return it!
4. That’s it!!!

THE FUNCTION

```
String reverse(String word){
    if(word.length()==1)
        return word;→BASE CASE!
    else
        return (String)word.charAt(word.length()-
    1)+reverse(word.substr(0,word.length()-1));
}
```

RECURSIVE LINEAR SEARCH

- How can we search and see if an item exists in an array recursively?
- To do this we need to think about how we can apply a divide and conquer strategy on the array?

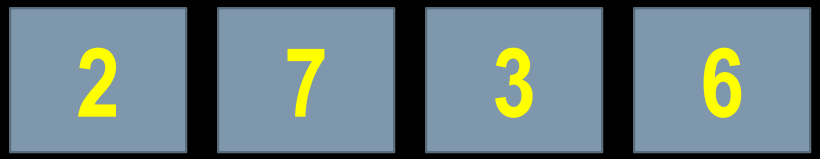
RECURSIVE LINEAR SEARCH

As an example lets look at the following array:

Let's assume we're looking for 3.

Let's start by checking the first element and see if it matches.

Of course it doesn't so we can remove that element.



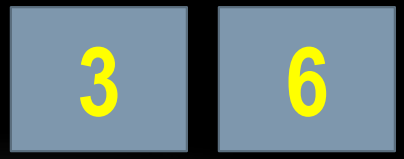
RECURSIVE LINEAR SEARCH

Let's start by checking the first element and see if it matches.

Of course it doesn't so we can remove that element.

Let's start by checking the first element and see if it matches.

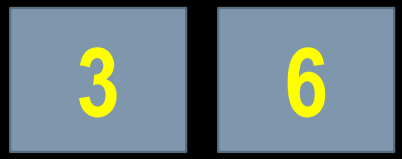
Of course it doesn't so we can remove that element.



RECURSIVE LINEAR SEARCH

Let's start by checking the first element and see if it matches.

Of course it DOES so return true (BASE CASE)



What if what we're searching for doesn't exist at all. What's the base case then?

If the length of the array is 0 then return false!

RECURSIVE LINEAR SEARCH-PSEUDOCODE

```
boolean search(ArrayList arr, int item){
        if(arr.length==0)
                return false;
        else if(arr[0]==item)
                return true;
        else
                return search(arr.remove(0),item);
}
```

EXERCISES

- 1. Write and test a recursive function max to find the largest number in a list(array of numbers).
- 2. Write a recursive function that detects whether a word is a palindrome.
- 3. Given a non-negative int n, return the sum of its digits recursively (no loops). Note that mod (%) by 10 yields the rightmost digit (126 % 10 is 6), while divide (/) by 10 removes the rightmost digit (126 / 10 is 12).

sumDigits(126) → 9
sumDigits(49) → 13
sumDigits(12) → 3

- 4. What is the output of the following recursive program? print(gcd(20, 12));

```
int gcd(a, b)
{
        if b == 0{ return a}
    else
        return gcd(b, a % b);
}
```

- 5. Write a recursive function that finds and returns the minimum element in an array, where the array and its size are given as parameters.
- //return the minimum element in a[] int findmin(int a[], int n)

- 6. Write a recursive function that computes and returns the sum of all elements in an array, where the array and its size are given as parameters.
- //return the sum of all elements in a[] int findsum(int a[], int n)

- 7. Given a string, compute recursively a new string where all the adjacent chars are now separated by a "*".

  allStar("hello") → "h*e*l*l*o"
  allStar("abc") → "a*b*c"
  allStar("ab") → "a*b"

- 8. Given an array of ints, compute recursively the number of times that the value 11 appears in the array.

  array11({1, 2, 11}, 0) → 1
  array11({11, 11}, 0) → 2
  array11({1, 2, 3, 4}, 0) → 0

Once you have answered all your questions on this ppt, post it to your Unit 3 web page.

Make sure you have tested your solutions either by writing and running them through code or by manual trace. You will be given an assignment soon to test your knowledge.