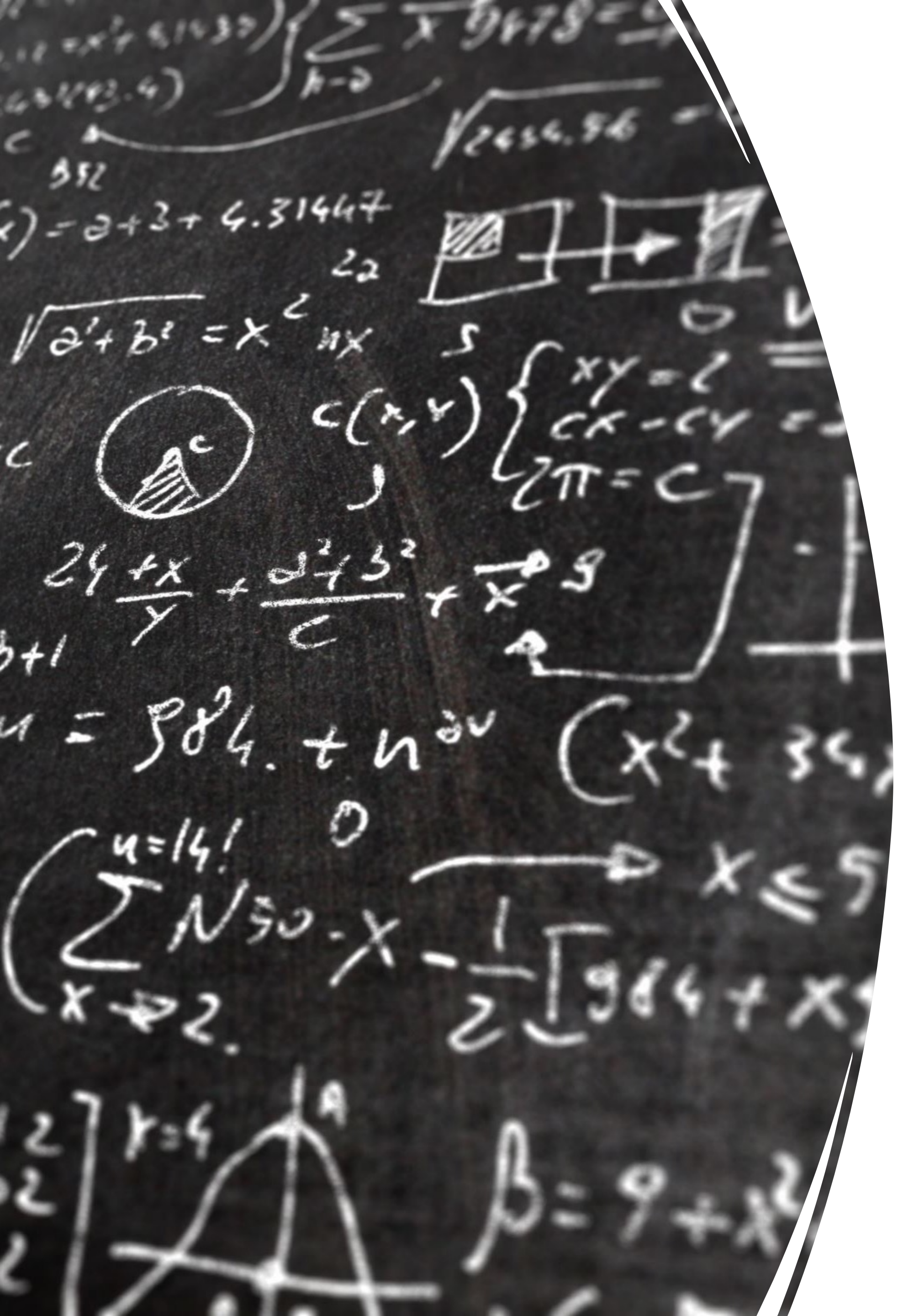

CS 113 – Computer Science I

Lecture 20 –Recursion

Adam Poliak

11/22/2022

Announcements

- Assignment 09
  - Due Wednesday 11/23

- No code jam this week during lab

Recursion

An **iteratively** algorithm uses a loop to perform repetition

Recursion - a function that calls itself

Conceptually like a loop (code repeats)

Easier way to solve “similar” problems

Creating a recursive algorithms

Rule that “does work” then ”calls itself” on a smaller version of the problem

Base case that handles the smallest problem

Prevents “infinite recursion”

Recursion example - tower

Draw a tower with height 6 blocks

Rule: Place one block and then draw a tower slightly shorter

Base case: When the height is 0 draw nothing

Recursion example – print “hello” 5 times

Rule: Print “hello” once and then print “hello” 4 times

Base case: When the number of times to print is 0, stop printing

Recursive functions – base case

Conditional statement that prevents infinite repetitions

Usually handles cases where:

input is empty

problem is at its smallest size

Recursion Example - Factorial

$$n! = n\ * (n-1) * (n-2) *\ \ldots * 1$$

3! = 3 * 2 * 1 = 6

4! = 4 * 3 * 2 * 1 = 24

Visualizing recursion – Factorial example

factorial(5) =

= 5 * factorial(4)

= 5 * 4 * factorial(3)

= 5 * 4 * 3 * factorial(2)

= 5 * 4 * 3 * 2 * factorial(1)

= 5 * 4 * 3 * 2 * 1

Recursion Example – Contains letter

Recursion Visualization – Contains letter

```
contains(“l”, “apple”) =
    contains(“l”, “apple”, 0)
        contains(“l”, “apple”, 1)
            contains(“l”, “apple”, 2)
                contains(“l”, “apple”, 3)
                    return true
```

Recursion Example – printList

Write a recursive function that prints the contents of an array

Recursion limitations

- Limited number of times we can recurse
    - Stackoverflow – too many frames

- Potentially memory inefficient
    - If we copy data in subproblems

- Performance: might duplicate unnecessary work