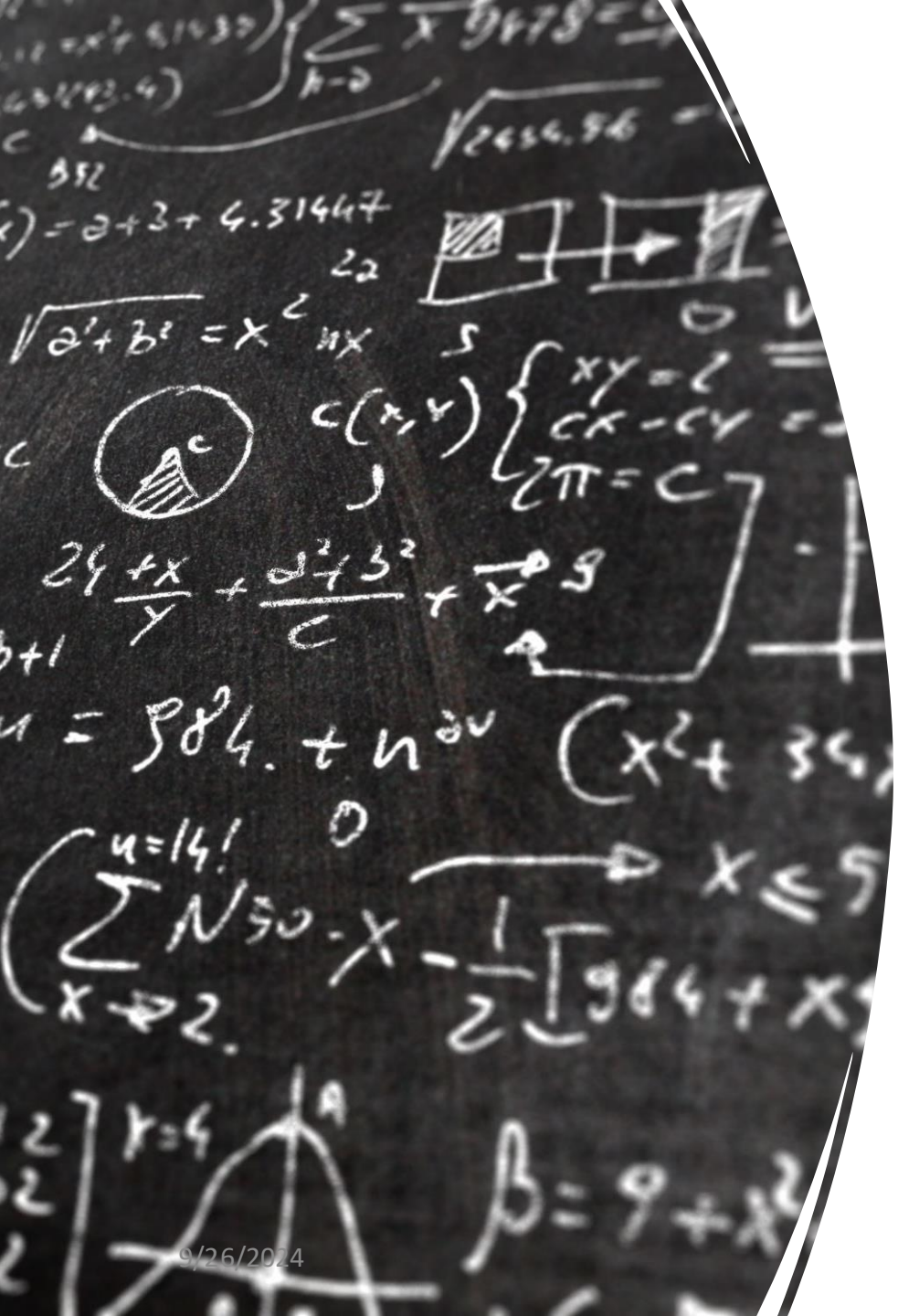

# CS 113 – Computer Science I

# Lecture 07 – Recursion

Tuesday 09/26/2024

# Announcements

- HW03 – released
  - Due Monday 09/30 11:59pm
- Thursday 10/03
  - Rosh Hashana
  - Either no class or guest lecture, TBD
- Office hours:
  - Adam’s 2:40-4:00pm today

# HW02 feedback

Lessons learned – emphasis on asking questions – great!

**How to ask for help:**

- Explain what you are trying to do
  - Give a minimal example
- Someone else should be able to replicate the problem easily
- Shouldn’t require any data/information that only you have
  - Explain what you think should happen
  - Explain what you get instead (copy / paste or screenshot if you can)
- Explain what else you’ve tried

# HW03

Implementing a bunch of String related methods

Start by writing method stubs

Example: *Write a method called isAbecedarian that takes a String and returns a boolean indicating whether the word is abecedarian.*

Upload method stubs to gradescope

# HW03

The autograder failed to execute correctly. Please ensure that your submission is valid. Contact your course staff for help in debugging this issue. Make sure to include a link to this page so that they can help you most effectively.

```
error: cannot find symbol assertTrue("isAbecedarian("abdest") should return true"
```

```
: error: cannot find symbol assertFalse("isDoubloon("baddeb") should return false"
```

# Agenda

Recursion

# Exercise: Blackjack

Write a program Blackjack.java which generates a random value between 2 and 21

- If the value is 21, print the value and “Blackjack” to the console
- If the value is between 17 and 20, print the value and “Stand” to the console
- If the value is less than 17, print the value and “Hit me!” to the console

# Top down design

1. Identify features of the program
    1. List them out!
2. Identify verbs and nouns in feature list
    1. Verbs: functions
    2. Nouns: objects/variables
3. Sketch major steps – how features should fit together
    1. Algorithm!
4. Write program skeleton
    1. Include function **stubs** (placeholders for our functions)
    2. Function **stub:** empty function with parameters and return type
5. Implement and test function stubs one at a time

# Recursion

# Recursion

a function that calls itself

“Simple” 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 – 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

Write a method called “containsLetter” that determines if a String contains a given character

Question: What are the parameters?

1. The String to be looking in
2. The character to look for

Question: What is the return type?

# Recursion Example – Contains letter

How can we break this problem down into smaller problems?

contains(“l”, “apple”) =

contains(“l”, “a”) OR

contains(“l”, “p”) OR

contains(“l”, “p”) OR

contains(“l”, “l”) OR

contains(“l”, “e”) OR

# Recursion Visualization – Contains letter

contains(“l”, “apple”) =

contains(“l”, “apple”)

contains(“l”, “pple”)

contains(“l”, “ple”)

contains(“l”, “le”)

return true

# Recursion Example – IndexOf letter

Write a method called IndexOf.

Arguments: String (haystack), Character (needle)

Return: the index of the character in the String, if the chatacter isnt there, return:

-1.

# Recursion Example – printVowels

Write a recursive function that prints just the vowels in a String

# Recursion limitations

- Limited number of times we can recurse
  - Stackoverflow – too many frames
- Potentially memory inefficient
  - If we copy data in subproblems – we'll worry about this in a few weeks
- Performance: might duplicate unnecessary work
  - We'll define performance later in the semester

# Style

- How we format our programs is **very** important
  - Like rules of etiquette around eating and keep a clean appearance
  - Like punctuation rules, it helps make text more readable
- Variable names should be descriptive
- Indentation is **very** important
  - Every statement inside a pair of braces must be indented
- Braces should be placed consistently