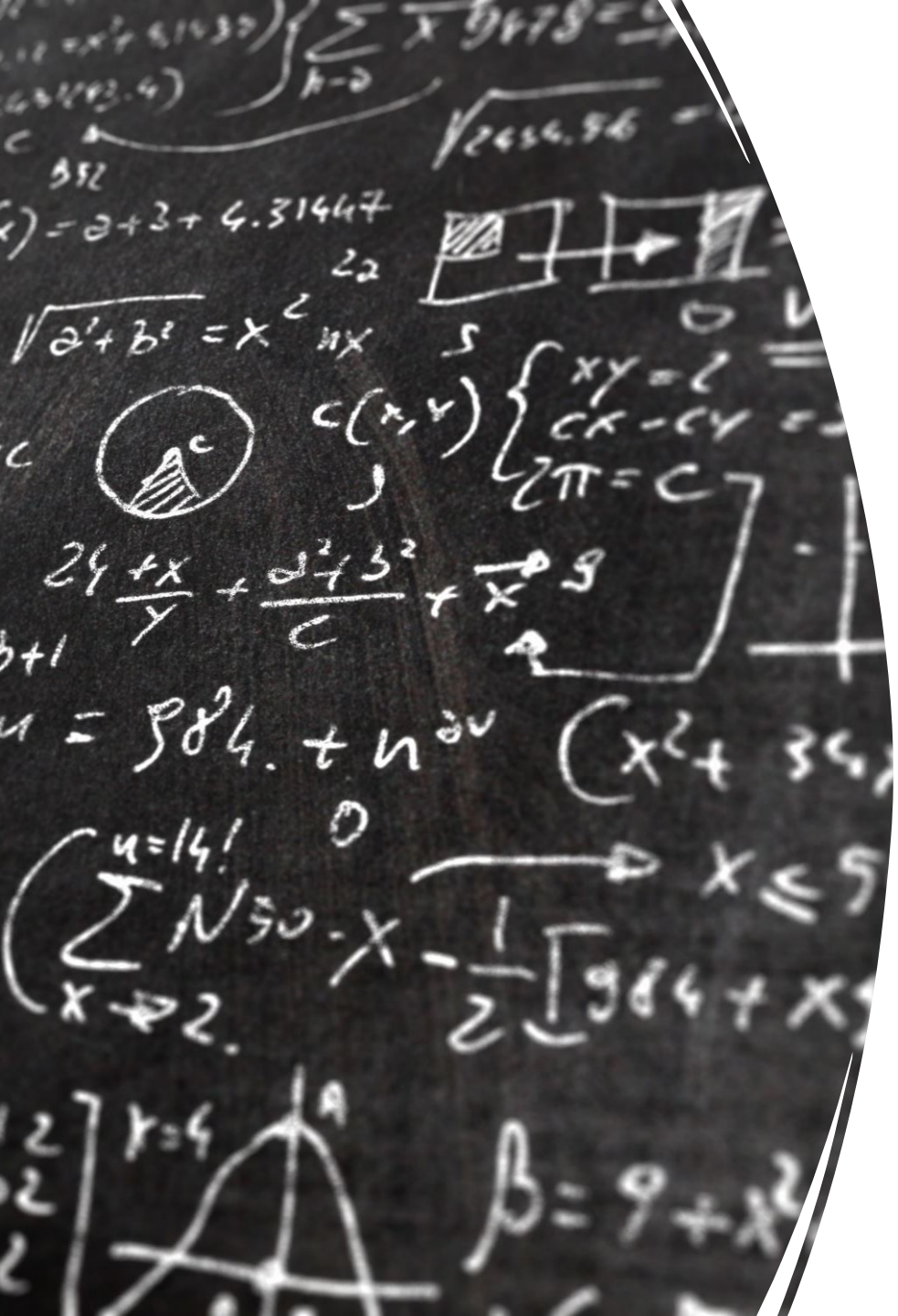


CS 113 – Computer Science I

Lecture 9 - Recursion

Adam Poliak

02/14/2023

Announcements

- HW03 – due tomorrow night (Wed 02/15)
- HW04 - released tonight
  - Due Tuesday 02/21
- Midterm in class Thursday 03/02
  - Closed book

Course Policies

Academic Integrity in Computer Science

**Discussing ideas and approaches to problems with others on a general level is fine (in fact, we encourage you to discuss general strategies with each other), but you should never read anyone else's code or let anyone else read your code.*

- All code you submit should be your own with the following permissible exceptions: code distributed in class, and code found in the course text book. In these cases, you should always include detailed comments that indicates on which parts of the assignment you received help, and what your sources were.
- Please don't hesitate to ask the awesome teaching assistants (TAs) for help. They provide TA hours most week nights and are excellent mentors!
- Please discuss the readings and associated topics with each other. Work together to understand the material. Reading groups to discuss the material are highly recommended —we will explore many ideas and it helps to have multiple people working together to understand them.
- It is fine to discuss the topics covered in the homeworks, to discuss approaches to problems, and to sketch out general solutions. However, you MUST write up the homework answers, solutions, and programs individually without sharing specific details, mathematical results, program code, etc.
- Under NO circumstances should you share computer code with another student. Similarly, you are **not permitted to use code found on the internet for any of your assignments**.
- Exams, of course, must be your own individual work.

Penalties

- 1st infraction: 0 on the duplicated code
- 2nd infraction: 0 on the assignment and email to dean
- 3rd infraction: honor board

Agenda

- Announcements
- Recap
- Recursion

Arrays

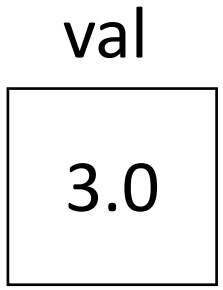
Idea: Store multiple values into a single variable

Values are sequential

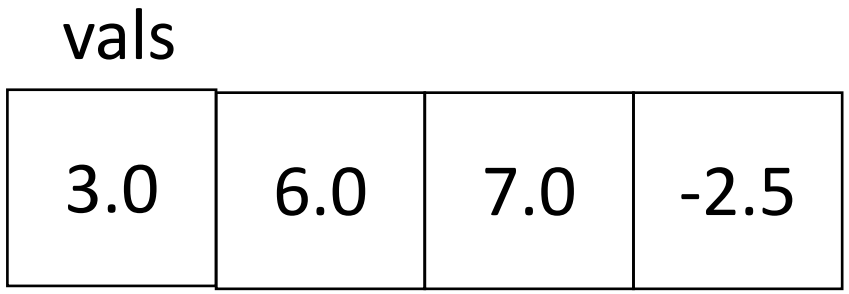
Analogous to a list

Arrays

```
double val = 3.0;
```




```
double[] vals = {3.0, 6.0, 7.0, -2.5};
```



Arrays

Three ways to initialize an array

1. With an initial value

```
int[] numbers = {1, 2, 5};
```

2. With allocated space, but uninitialized

```
int[] numbers = new int[3];
```

3. With an empty array reference

```
int[] numbers = null;
```

Array Indexing

Access individual elements of an array with indexing

array[index]

We use *zero*-based indexing

first element is **0**

last element is **length-1**

Accessing indices out of range results in a **runtime error**!

Strings

Strings are implemented as *arrays of characters*

Get the length of a string with length()

```
String greeting = “hola”;
int len = greeting.length(); // what is the length?
char c = greeting[2]; // what character is in index 2?
```

char: New built-in type, denoted with single quote, e.g. ‘a’ or ‘{‘

Exercise: GetCharacters.java

Write a program, GetCharacters.java, that asks the user for a word and then prints the first, last and middle character.

```
Enter a word: hola!
FirstIndex: 0 FirstCharacter: h
MiddleIndex: 2 MiddleCharacter: l
LastIndex: 5 LastCharacter: !
```

Command line arguments

```
public static void main(String[] args)
```

Command line arguments are an *array of String*

Exercise: Write a program called commandLineArgs.java that

1) prints out 3 command line arguments that are passed in.
2) Compute the sum of three command line arguments (assuming they are integers)

Agenda

- Announcements
- Recap
- Arrays
- **Recursion**

Washing dishes

Smart way to wash dishes

Punt the problem to someone else

But we want to wash one dish so we can say we washed a dish

Motivation #2 - adding

I’m going to give you a list of numbers

- Group A: each person adds up all the numbers
- Group B: one person takes the first number and passes the rest to the next person, repeat this process until no more numbers
  - Last person adds the last two numbers and send the result to the previous person
    - Who adds their number with the result …

Motivation #2 adding numbers

- 20,
- 43,
- 13,
- 13,
- 10,
- 43,
- 90,
- 32,
- 42

Motivation #2 adding numbers

- Which was easier?
- Which was like ”smarter” washing dishes?
  - How so?
- This is called 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

Recursion Visualization – Contains letter

contains(“l”, “apple”) =

contains(“l”, “apple”)

contains(“l”, “pple”)

contains(“l”, “ple”)

contains(“l”, “le”, 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 – we'll worry about this in a few weeks
- Performance: might duplicate unnecessary work
  - We'll define performance later in the semester