I'll Miss You
RecursionError and a limitation of recursion
Recall the recursive assumption

In the recursive step, assume that each recursive call works correctly.

```python
nested_sum(sublist)
subtree.insert(item)
left.__contains__(item)
quicksort(smaller)
```
But this assumption has a condition!

In the recursive step, assume that each recursive call works correctly...

*as long as the input(s) to the recursive call are smaller than the original(s)*
Danger: Infinite recursion

Infinite recursion occurs when a series of recursive calls never reaches a base case. Every call makes a new recursive call, and this continues forever!

```python
class Tree:
    def __len__(self) -> int:
        if self.is_empty():
            return 0
        else:
            return self.__len__()
```
Danger: infinite recursion

Because each function call requires a new stack frame, infinite recursion can lead to an "infinite" amount of computer memory consumption.

Python protects against this by limiting the total number of stack frames it allows.
RecursionError

Traceback (most recent call last):
    t.__len__()
    return self.__len__()
    return self.__len__()
    return self.__len__()
[Previous line repeated 994 more times]
    if self.is_empty():
RecursionError: maximum recursion depth exceeded
Setting the recursion limit

```python
import sys
sys.setrecursionlimit(5000)
```
(Brief) comments on the final exam
To be posted soon on course website

Some content details (short version: study everything)
Final exam aid sheet
Extra office hour schedule
Final exam study tips

1. **Concept mastery**: definitions, examples, connections
2. **Code mastery**: make a plan, break down tasks into small chunks, use code templates
3. **Practice in a test-like environment**: pencil and paper, closed book (+ aid sheet), timed environment
Aim for mastery

“Mastery” means that you are the expert.

Go beyond “re-reading and re-doing”:
- create multiple solutions
- identify common mistakes or errors, and
- explain ideas to others
- make up new questions
Hey folks,

Anybody want to study for the final together? I’m going to be in BA3200 on _____ at 1pm onwards, feel free to join! I’m looking to do really well on the final, so some serious studying will be happening. I’ll be the one wearing the AWESOME cat sweater.

P.S. This post is so ironic because I copied exactly what David showed us in class, lol.
Looking back, looking ahead

WHAT WERE THE BIG THEMES FROM CSC148?
WHERE CAN YOU GO FROM HERE?
You are no longer a novice programmer.

You have written significant Python code and covered a broad range of practical experience, including:

- Working with code across multiple files and classes
- Using code-writing tools like PyCharm and python_ta
- Testing and profiling tools (pytest, hypothesis, timeit)
- Useful Python libraries (csv, random, pygame)
Using Python in the "real world"

You can apply your skills to...

- analyze and present data (Python Data Science Handbook)
- automate computational tasks (Automate the Boring Stuff)
- create websites and web applications (Full Stack Python)
- ...and more!
Software engineering principles

How do we design software that is modular, maintainable, and robust against failure?

How do we do this while working on a team of people?

- CSC207: Software Design
- CSC301: Introduction to Software Engineering
- CSC302: Engineering Large Software Systems
From software to hardware

What does it mean to interact with computer memory directly (like in the C programming language)?

What actually happens when we "run" a program?

- CSC209: Software Tools and Systems Programming
- CSC258: Computer Organization
- CSC369: Operating Systems
Data Structures and algorithms

Given a problem to solve or set of operations to implement, how do we design algorithms and data structures for them?

How do we analyze the correctness and efficiency of our data structures and algorithms?

- CSC236: Introduction to Theory of Computation
- CSC263: Data Structures and Analysis
- CSC373: Algorithm Design, Analysis, and Complexity
Upper-year courses

**Artificial intelligence:** CSC384, CSC411, CSC412, CSC421, CSC486

**Systems and networks:** CSC358, CSC367, CSC369, CSC459, CSC469

**Theory of computation:** CSC438, CSC448, CSC463, CSC473

**Human-computer interaction:** CSC318, CSC418

**Scientific computing:** CSC336, CSC436, CSC446, CSC456, CSC466

**Computer graphics and vision:** CSC320, CSC418, CSC420

**Programming language design and implementation:** CSC324, CSC410, CSC488

...and more!
Co-curricular opportunities

Undergraduate research:
http://web.cs.toronto.edu/program/ugrad/research.htm

Professional Experience Year:
http://web.cs.toronto.edu/program/ugrad/pey_co-op.htm

Computer Science Student Union
https://cssu.ca/
A story in four quotations

David: There's a word that starts with “C” that's very important in this course.

Student: ...computer?

David: 💔
Code aesthetics is communication

“Programs must be written for people to read, and only incidentally for machines to execute.”

Harold Abelson
Documentation is communication

“The most important single aspect of software development is to be clear about what you are trying to build.”

Bjarne Stroustrup
Software design is communication

“There are two ways of constructing a software design: One way is to make it so simple that there are obviously no deficiencies, and the other way is to make it so complicated that there are no obvious deficiencies. The first method is far more difficult.”

Sir Charles Antony Richard Hoare
A Case Study: David's projects
Thank you for being a great class!

Good luck on the final exam, and have a restful holiday break!