Computational Structures in Data Science

Wrap Up

Week 7, Summer 2024. 8/1 (Thurs)

Lecture 25

Berkeley
UNIVERSITY OF CALIFORNIA

Announcements

- "Official" course evaluation survey released
  - https://course-evaluations.berkeley.edu/Berkeley/
  - Filling this out will be a huge help for the Data/DSUS department with regard to course development, so please fill this out :)
- Project02 ("Ants") due tonight (8/1)!
- Ed post: "What's Left in C88C?"
- Reminder: "clobber" policy for final exam vs midterm

Announcements: Final Exam

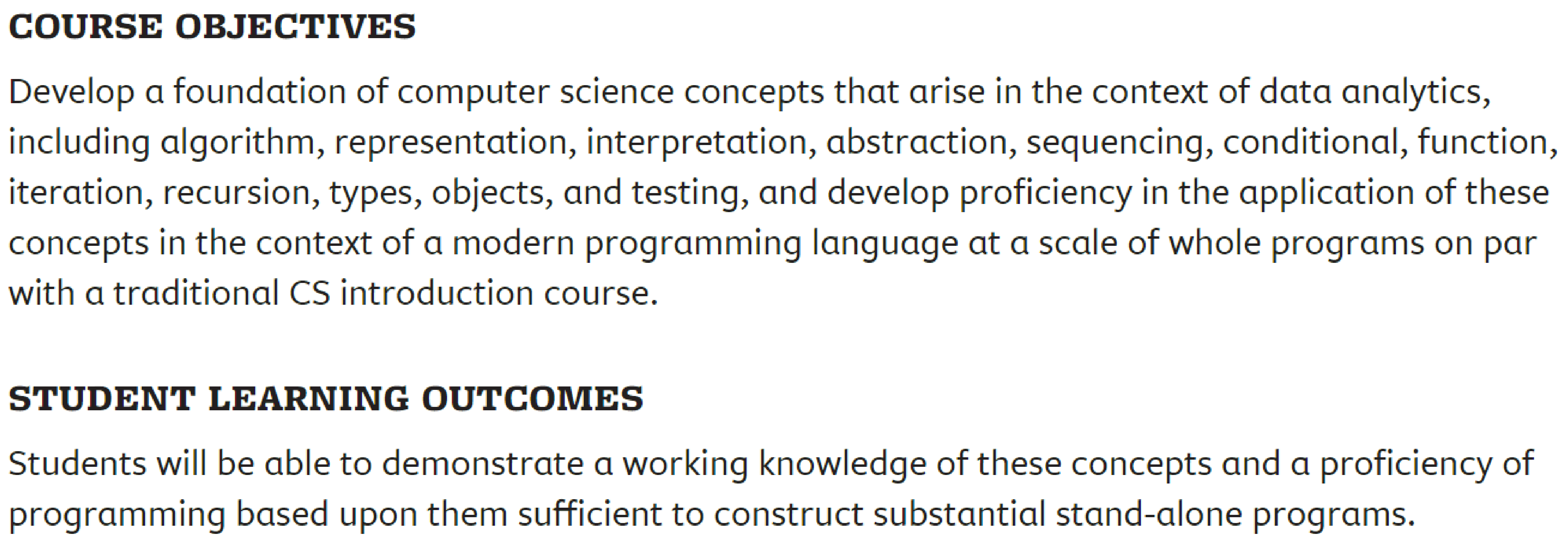
- Final exam "primary" time: Wednesday August 7th, 3PM – 5 PM PST
- Final exam scheduling form released
  - Gradescope: "(Urgent) Final Exam Scheduling"
  - Due: Wednesday July 31st 11:59 PM PST
  - Please fill this out ASAP! For more info, see Ed post.
    - DSP students: you should have received an email for a separate Final exam scheduling form. Please fill this out – if you didn't receive an e-mail, let us know!
- Final exam will have the same style as the Midterm
  - Zoom+Gradescope

Is Your Brain Full Yet?

- Data: values, literals, operations,
- Functions
- Variables
- List, Tuples, Dictionaries
- Function Definition Statement
- Conditional Statement
- Iteration: list comp, for, while
- Lambda function expr.
- Higher Order Functions
- Higher order function patterns
  - Map, Filter, Reduce
- Recursion
- Abstract Data Types
- Mutation
- Class & Inheritance
- Exceptions
- SQL / Declarative Programming

Data C88C: Course objectives

Data C88C su24: https://classes.berkeley.edu/content/2024-summer-data-c88c-001-lec-001



COURSE OBJECTIVES

Develop a foundation of computer science concepts that arise in the context of data analytics, including algorithm, representation, interpretation, abstraction, sequencing, conditional, function, iteration, recursion, types, objects, and testing, and develop proficiency in the application of these concepts in the context of a modern programming language at a scale of whole programs on par with a traditional CS introduction course.

STUDENT LEARNING OUTCOMES

Students will be able to demonstrate a working knowledge of these concepts and a proficiency of programming based upon them sufficient to construct substantial stand-alone programs.

LOTS of Courses to Follow Up! [Courses]

- Explore all the DS connectors
  - Data 88E (Econ)
  - Stat 88 / DATA C88S
- Data 100: Principles of DS
- Data 101: Data Engineering
- Data 102: Data, Inference, and Decisions
- Data 104: Human Context and Ethics
- Data 140: Probability
- INFO – Course Schedule Varies!
  - INFO C103 in Spring: "History of Information"
  - Some Database courses, web development, etc.

CS Courses

- CS61B: (conventional) data structures, statically typed production languages.
- CS61C: computing architecture and hardware as programmers see it.
- CS70: Discrete Math and Probability Theory.
- CS170, CS171, CS172, CS174: “Theory”—analysis and construction of algorithms, cryptography, computability, complexity, combinatorics, use of probabilistic algorithms
- CS161: Security
- CS162: Operating systems.
- CS164: Implementation of programming languages
- CS168: Introduction to the Internet
- CS160, CS169: User interfaces, software engineering
- CS176: Computational Biology

CS Courses Part 2

* CS182, CS188, CS189: Neural networks, Artificial intelligence, Machine Learning
- CS184: Graphics
- CS186: Databases
- CS191: Quantum Computing
- CS195: Social Implications of Computing
- EECS 16A, 16B: Designing Information Systems and Devices
- EECS 126: Probability and Random Processes
- EECS149: Embedded Systems
- EECS 151: Digital Design
- CS194: Special topics. (e.g.) computational photography and image manipulation, cryptography, cyberwar.
- Plus graduate courses on these subjects and more.

EE Courses Are There Too

- EE105: Microelectronic Devices and Circuits.
- EE106: Robotics
- EE118, EE134: Optical Engineering, Photovotalaic Devices.
- EE120: Signals and Systems.
- EE123: Digital Signal Processing.
- EE126: Probability and Random Processes.
- EE130: Integrated Circuit Devices.
- EE137A: Power Circuits.
- EE140: Linear Integrated Circuits (analog circuits, amplifiers).
- EE142: Integrated Circuits for Communication.
- EE143: Microfabrication Technology.
- EE147: Micromechanical Systems (MEMS).
- EE192: Mechatronic Design.

Eric Kim's favorite classes (in no particular order)

- CS 61BL: sparked my love of CS and programming, taught by Prof. Colleen Lewis (great prof!)
- CS 188, 189: my first AI class, taught by Prof. Klein, Prof. Russell (great profs!)
- CS 161 (Computer Security): really, really fun class taught by Prof. David Wagner (great prof!)
- CS 164 (Programming Languages and Compilers): Taught by Prof. Paul Hilfinger (great prof!).
- History 124B ("The Recent United States: The United States from World War II to the Vietnam Era"): really great professor (Dr. Kathleen J. Frydl) that opened my eyes to, in my opinion, one of the most interesting + impactful (to today's modern world) parts of history.
- Music 60, 61 ("Harmony"): Really enjoyable, memorable adventure into functional harmony, taught by Prof. David Pereira (great prof!).

And there’s lots more to Python!

What can you do with Python?

- Almost anything!
- Webapp backends
- Web scraping
- Natural Language Processing
- Data analysis
- Machine Learning
- Scientific computing
- Games

What can you do with Python?

- Almost anything! Thanks to libraries!
- Don't try to memorize libraries, learn how to learn how to use them!
- Webapp backends (Flask, Django)
- Web scraping (BeautifulSoup)
- Natural Language Processing (NLTK)
- Data analysis (Numpy, Pandas, Matplotlib)
- Machine Learning (FastAi, PyTorch, Keras, Tensorflow)
- Scientific computing (SciPy)
- Games (Pygame)

DS/CS is ubiquitous in today's world

- The skills you'll learn in this course, and in the rest of your DS/CS degree, will be invaluable to MANY disciplines and industry+academia
  - Anecdote: in medicine+tech, it's easier for a DS/CS person to pick up relevant medical knowledge than it is for a doctor to pick up relevant programming/data-science knowledge.
- Intersection of X + CS/DS is really exciting! Take advantage!
  - Health/medicine + CS/DS
  - Finance + CS/DS
  - Entertainment + CS/DS
  - Law + CS/DS
  - (the sky is the limit!)

Computational Structures in Data Science

Final Review

Approaching The Exam

- Skim the topics (~1 min)
- Handle the "easy"(est) questions first
- Read the whole question first!
- Read the text
- Read the doctests!
- What techniques might be applicable?
  - Pattern matching is OK
- Draft a solution on scratch paper!
- Write yourself notes

SQL practice: https://code.cs61a.org

- For following SQL practice, we will use the `records`, `salaries` table from code.cs61a.org
- # (Question) what SQL query to show, for each employee's row, their supervisor's salary? (2022 salary is fine)

-
  ```
  select records.name, records.salary,
  records.supervisor, records2.salary as salarySupervisor
  from records, records as records2 where
  records.supervisor = records2.name;
  ```

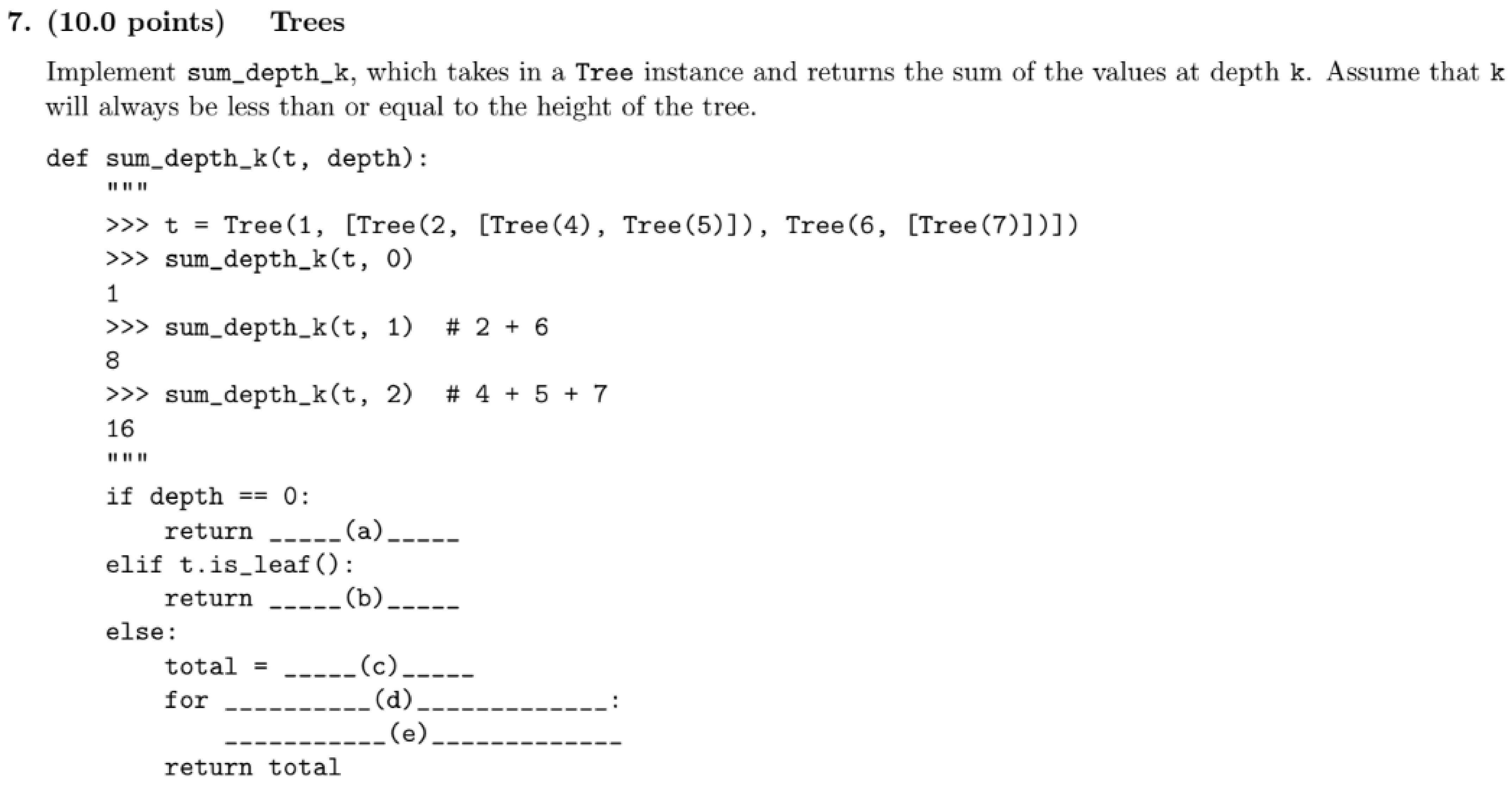
SQL practice: https://code.cs61a.org

- For following SQL practice, we will use the `records`, `salaries` table from code.cs61a.org
- # (Question) which SQL query to show the salary increases from 2022 -> 2023?
- `select name, salary2023 - salary2022 as salaryDelta2022to2023 from salaries;`

SQL practice: https://code.cs61a.org

- For following SQL practice, we will use the `records`, `salaries` table from code.cs61a.org
- # (Question) which SQL query to show each department's min, max, and avg salary?
- `select division, MIN(salary) as minSalary, MAX(salary) as maxSalary, AVG(salary) as avgSalary from records group by division;`

Spring 2024 – Question 7 - Trees



7. (10.0 points) Trees

Implement `sum_depth_k`, which takes in a `Tree` instance and returns the sum of the values at depth `k`. Assume that `k` will always be less than or equal to the height of the tree.

```
def sum_depth_k(t, depth):
    """
    >>> t = Tree(1, [Tree(2, [Tree(4), Tree(5)]), Tree(6, [Tree(7)])])
    >>> sum_depth_k(t, 0)
    1
    >>> sum_depth_k(t, 1)  # 2 + 6
    8
    >>> sum_depth_k(t, 2)  # 4 + 5 + 7
    16
    """
    if depth == 0:
        return _____(a)_____
    elif t.is_leaf():
        return _____(b)_____
    else:
        total = _____(c)_____
        for __________(d)_____________:
            ___________(e)_____________
        return total
```

Summer 2024 Midterm: Env Diagram

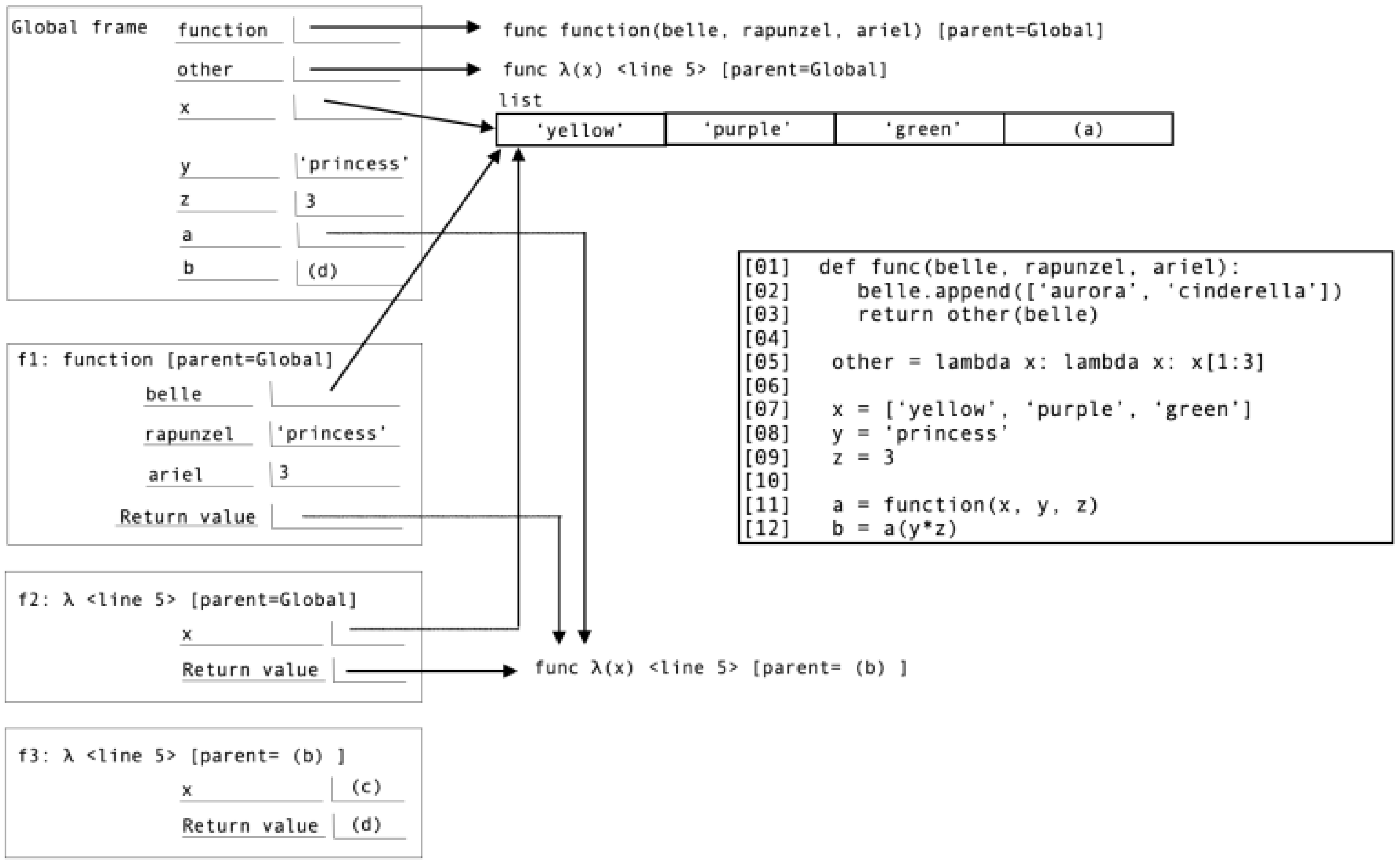
Fill in the blanks to complete the environment diagram. All the code used is in the box to the right, and the code runs to completion.

```
def function(belle, rapunzel, ariel):
    belle.append(['aurora', 'cinderella'])
    return other(belle)

other = lambda x: lambda x: x[1:3]

x = ['yellow', 'purple', 'green']
y = 'princess'
z = 3

a = function(x, y, z)
b = a(y * z)
```



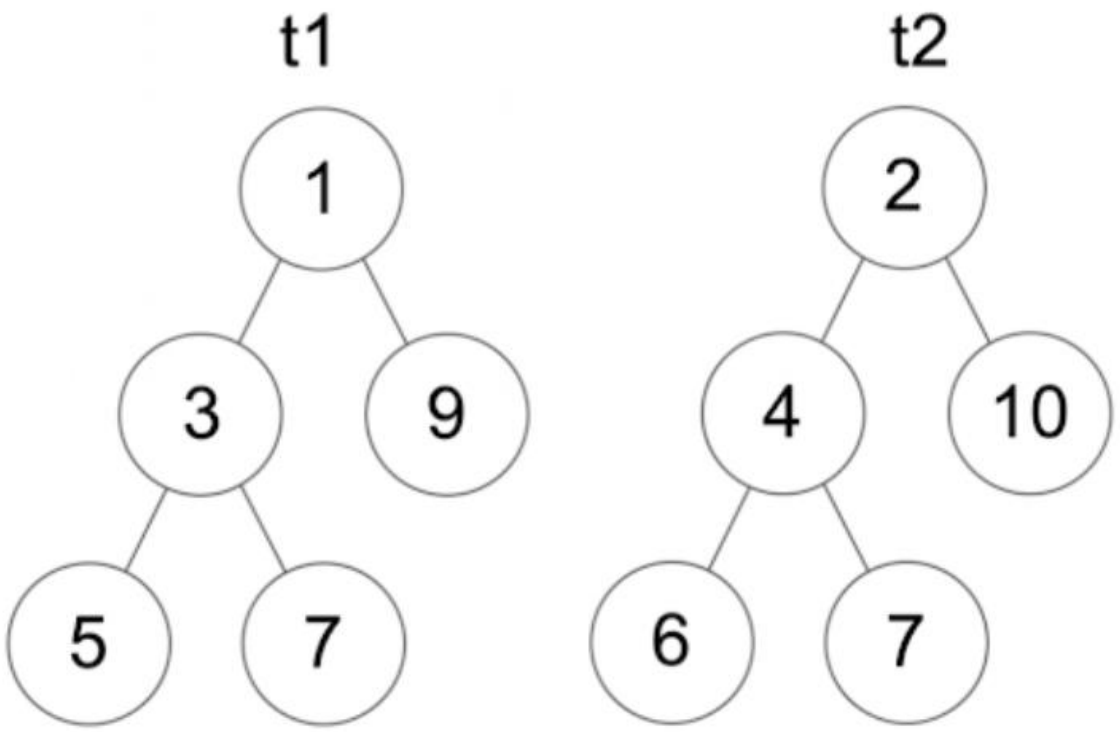
Global frame
function
other
x
y
'princess'
z
3
a
b
(d)
func function(belle, rapunzel, ariel) [parent=Global]
func λ(x) <line 5> [parent=Global]
list
'yellow'
'purple'
'green'
(a)
f1: function [parent=Global]
belle
rapunzel
'princess'
ariel
3
Return value
f2: λ <line 5> [parent=Global]
x
Return value
func λ(x) <line 5> [parent= (b)]
f3: λ <line 5> [parent= (b)]
x
(c)
Return value
(d)

```
[01]   def func(belle, rapunzel, ariel):
[02]      belle.append(['aurora', 'cinderella'])
[03]      return other(belle)
[04]
[05]    other = lambda x: lambda x: x[1:3]
[06]
[07]    x = ['yellow', 'purple', 'green']
[08]    y = 'princess'
[09]    z = 3
[10]
[11]    a = function(x, y, z)
[12]    b = a(y*z)
```

Fall 2021 Q9 – Tree Farm

9. (12.0 points) Tree Farm

You've decided to get into the tree growing business! All the trees you grow have the same structure as each other but may have different values. You want to detect the nodes that are in the same position in two given trees but have different values. Write a function that takes in two trees, `t1` and `t2`, with the same structure and yields the mismatching node values as a tuple.



THANK YOU!

COME JOIN COURSE STAFF!

Computational Structures in Data Science

THANK YOU!

(Again!)

Berkeley
UNIVERSITY OF CALIFORNIA