Announcements

- Homework 12 due Tuesday 12/10 @ 11:59pm.
- All you have to do is vote on your favorite recursive art.
- 29 review sessions next week! Come learn about the topics that interest you the most.
- See http://inst.eecs.berkeley.edu/~cs61a/fa13/exams/final.html for the schedule.
- The final exam is on Friday 12/20 @ 11:30am in the RSF gym, emphasizing:
  - Higher-order functions
  - Sequences (tuples, lists, recursive lists, Scheme lists)
  - Non-local assignment and mutation
  - Object-oriented programming
  - Recursion and recursive data
  - Iterators, generators, and streams

Example: Numerical Approximations

No calculators/interpreters allowed!
Let’s say we have a computer that can +, -, *, /.
How do we answer this question?

(A) A sequence of approximations (SoA) to \( \pi \) is an infinite sequence that converges to \( \pi \).

Implicitly define a SoA to \( \pi \).

```python
def sqrt(a):
    x = 1
    while True:
        yield x
        x = (x + a/x)/2
```

(H) Define a sequence of approximations to \( \pi \).

```python
def pi():
    total, k = 0, 1
    while True:
        yield total
        total += 8/(4*k-3)*(4*k-1)
        k += 1
```

Approximating Square Roots

Is \( \sqrt{2} - 4 < \pi \) ?
No calculators/interpreters allowed!
Let’s say we have a computer that can +, -, *, /.
How do we answer this question?

(A) A sequence of approximations (SoA) to \( \sqrt{2} \) is an infinite sequence that converges to \( \sqrt{2} \).

```python
def sqrt(a):
    x = 1
    while True:
        yield x
        x = (x + a/x)/2
```

Approximating Pi

Is \( \sqrt{3} - 4 < \pi \) ?
No calculators/interpreters allowed!
Let’s say we have a computer that can +, -, *, /.
How do we answer this question?

(H) Define a sequence of approximations to \( \pi \).

```python
def pi():
    total, k = 0, 1
    while True:
        yield total
        total += 8/(4*k-3)*(4*k-1)
        k += 1
```

Approximating Square Roots

Is \( \sqrt{2} - 4 < \pi \) ?
No calculators/interpreters allowed!
Let’s say we have a computer that can +, -, *, /.
How do we answer this question?

(A) A sequence of approximations (SoA) to \( \sqrt{2} \) is an infinite sequence that converges to \( \sqrt{2} \).

```python
def sqrt(a):
    x = 1
    while True:
        yield x
        x = (x + a/x)/2
```

Approximating Pi

Is \( \sqrt{3} - 4 < \pi \) ?
No calculators/interpreters allowed!
Let’s say we have a computer that can +, -, *, /.
How do we answer this question?

(H) Define a sequence of approximations to \( \pi \).

```python
def pi():
    total, k = 0, 1
    while True:
        yield total
        total += 8/(4*k-3)*(4*k-1)
        k += 1
```

Approximating Square Roots

Is \( \sqrt{2} - 4 < \pi \) ?
No calculators/interpreters allowed!
Let’s say we have a computer that can +, -, *, /.
How do we answer this question?

(A) A sequence of approximations (SoA) to \( \sqrt{2} \) is an infinite sequence that converges to \( \sqrt{2} \).

```python
def sqrt(a):
    x = 1
    while True:
        yield x
        x = (x + a/x)/2
```

Approximating Pi

Is \( \sqrt{3} - 4 < \pi \) ?
No calculators/interpreters allowed!
Let’s say we have a computer that can +, -, *, /.
How do we answer this question?

(H) Define a sequence of approximations to \( \pi \).

```python
def pi():
    total, k = 0, 1
    while True:
        yield total
        total += 8/(4*k-3)*(4*k-1)
        k += 1
```
(C) Assume that \( s \) is a SoA to \( y \) and each element of \( s \) is closer to \( y \) than the last. Define less_than_0(\( s \)) that returns True if it is certain that \( y < 0 \).

```python
def sqrt(a):
    x = 1
    while True:
        yield x
        x = (x + a / x) / 2

def pi():
    total, k = 0, 1
    while True:
        yield total
        total += 8 / ((4 * k - 3) * (4 * k - 1))
        k += 1

def four():
    while True:
        yield 4

def subtract(x, y):
    while True:
        yield next(x) - next(y)

def less_than_0(s):
    current = next(s)
    while True:
        last, current = current, next(s)
        if last < 0 and current < last:
            return True

a = subtract(sqrt(51), four())
less_than_0(subtract(a, pi()))
```

Computer Science

61A was Designed to Introduce the Big Ideas in Computer Science

- What are functions, data, sequences, trees, programs, languages, and interpreters.
- How to write legible programs, use recursion, measure complexity, and solve problems.
- Different programming paradigms: functional, object-oriented, and declarative.

What's left to learn in CS?
- Designing and testing software
- Algorithms for solving known problems
- Low-level representations of data and programs
- Discrete mathematics and analysis of programs
- Programming languages
- User interface design
- Networking
- Systems
- Artificial intelligence
- Lots of other subfields: graphics, theory, scientific computing, security, etc.

Life

Important Ideas Take a Long Time to Learn

- It's a good idea to study subjects other than computer science.
- Who you spend your time with is important.
- Ideas come from people, and people think from experience.
- Don't compare.
- Contribute to the world.

Thanks for being amazing!
Please stay for the HKN survey.