Announcements
Assignment 1 and Midterm marks to be released this afternoon.
Starting Monday, labs will be in BA3175 only.

Last time...
A choice of number between start and end is either start or a choice of number between start and end - 1.

```
(define (num-between start end)
  (if (equal? start (- end 1))
      (-< start)
      (-< start (num-between (+ 1 start) end))))
```

<- gives us a way of specifying what data we want to produce, independent of how it will be used.

“Declarative programming is a non-imperative style of programming in which programs describe their desired results without explicitly listing commands or steps that must be performed.” (Wikipedia)
Quick demo: converting choices into streams
(Homework: study implementation)

Backtracking search
Goal: find a value \( x \) that satisfies a predicate \( \text{pred} \).

1. Find a candidate value by making choices.
2. Check whether the current value satisfies \( \text{pred} \).
3. If not, \text{backtrack} and make different choices.

Query operator \( ?- \) (attempt 1)

```
(define (?- pred expr)
  (if (pred expr)
      expr
      DONE))
```
Query operator \texttt{?-} (attempt 2)

Automatically calling \texttt{(next!)}

\begin{verbatim}
(define (?- pred expr)
  (if (pred expr)
      expr
      (next!)))
\end{verbatim}

Problem:

\begin{verbatim}
> (* 100 (?- even? (-< 1 2 3 4 5)))
20000
\end{verbatim}
Query operator ?- (attempt 3)

Automatic backtracking

```
(define (?- pred expr)
  (if (pred expr)
      expr
      (backtrack!)))

(define (backtrack!)
  (shift k (next!)))
```

Applying backtracking search
**Prime sum**

Find pairs of numbers in $1, \ldots, n$ whose sum is prime.

**Satisfiability**

Find a satisfying assignment for a propositional formula.

$$(x_1 \lor x_2 \lor x_4) \land (x_2 \lor x_3 \lor x_4) \land (\overline{x_2} \lor \overline{x_3} \lor \overline{x_4}) \land$$

$$(\overline{x_1} \lor x_3 \lor \overline{x_4}) \land (x_1 \lor x_2 \lor x_3) \land (x_1 \lor x_2 \lor \overline{x_4})$$
Synthesizing arithmetic expressions

Find an arithmetic expression that evaluates to 10.

Generators
**generator**

an entity (in a program) that yields a sequence of values over time

---

In Racket: an entity that holds a private choices stack

```scheme
(define-generator (my-range start end)
        (define (num-between start end)
            (if (equal? start (- end 1))
                (< start)
                (< start (num-between (+ 1 start) end))))
        (num-between start end))
```

*my-range* returns a generator.
In Python: an entity that uses the `yield` operation

```python
def my_range(start, end):
    i = start
    while i < end:
        yield i
        i += 1
```

```python
>>> nums = my_range(10)
>>> nums
<generator object my_range at 0x...>
>>> nums.__next__()
10
>>> nums.__next__()
11
>>> nums.__next__()
12
```
```
def bisect(f, tol, a, b):
    # Precondition: f(a) and f(b) have different signs.
    c = (a + b) / 2
    while abs(f(c)) >= tol:
        if sign(f(a)) == sign(f(c)):
            a = c
        else:
            b = c
        c = (a + b) / 2
    return c
```
def bisections(f, a, b):
    # Precondition: f(a) and f(b) have different signs.
    while True:
        c = (a + b) / 2
        yield c
        if sign(f(a)) == sign(f(c)):
            a = c
        else:
            b = c