Strings in more depth

- Basically, vector<char>, but with a nicer name (string) + bonus functions.

- Bonus Functions:
  - concatenation: s1+s2 (s1 += s2 works as well)
  - "ctrl-f": s1.find(s2) will tell you if s2 appears in s1 (and where). (Return value tells you where match starts; -1 ≡ string::npos)
  - Not a bonus, but s.length() gives size of string s (# characters).
    s.size() works too (remember: strings are vectors)

    (I'm 99% sure s.length() is just a synonym for s.size().)

Exercises

Let's write our own version of find.

find(s1, s2) should return location (index) of first match in s1 of s2, or -1 if no match found.

s1.find(s2) ↑ STL version

find(s1, s2) ↑ our version

Ideas? Check for match @ every possible starting point.

Possible starting points? Say n1 = s1.length()
n2 = s2.length()

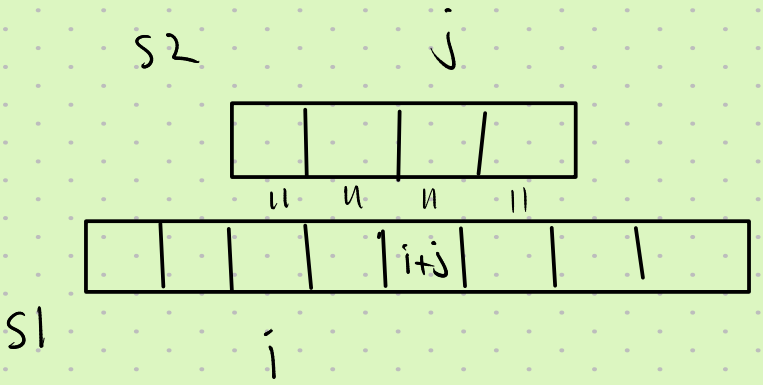
0, 1, 2, ...? if n1 == n2, only possibility is 0.

0, 1, ... n1-n2 ✓

Outline:
```
for (i=0; i<=n1-n2; i++) {
    // check for match @ i
    if (match(s1, s2, i)) return i;
```

```
    }
    return -1;
```

How to check for a match @ i?



for j = 0, 1, ... n2-1, must have

s2[j] == s1[i+j].

in s2:	in s1:
0	i
1	i+1
2	i+2
⋮	⋮