Strings (more detail)

String $\equiv$ vector<char>

+ renaming of a few functions.

vector	string
v.size()	s.length()
v.push_back(x)	s += x;

Bracket notation (s[i]) works for strings too.

Exercise: Write a function that reverses a string given as input. Reverse the string "in-place".

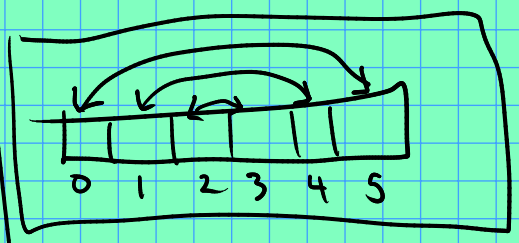
Simple not in-place solution:

```
void reverse(string& s)
{   string temp;
    for(int i = s.length()-1; i != -1; i--)
        temp += s[i];
    s = temp;
}
```

Works, but not "in-place". Requires 2x the memory of s.

In-place solution

```
void reverse (string & s)
{
  size_t len = s.length();
  for (size_t i = 0; i < len/2; i++){
    // swap s[i] <-> s[len-i-1]
    char temp = s[i];
    s[i] = s[len-1-i];
    s[len-1-i] = temp;
  }
}
```



$0 \leftrightarrow 5$

$1 \leftrightarrow 4$

$2 \leftrightarrow 3$

Generally,

$i \leftrightarrow len - i - 1$.

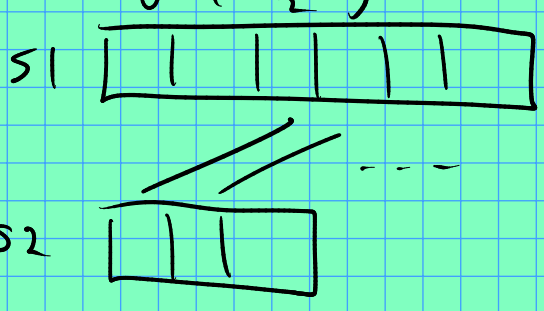
More challenging example: search for a substring.

Input: s1, s2. If s2 is a substring of s1, return the index of the first match.

Else Return -1.

```
size_t substr (const string & s1, const string & s2)
{
    ;
```

Idea:



Idea: test for match @ all possible starting positions.

Possible starting positions:

0, 1, ... (len1 - len2)

Outline:

```
for (i=0; i < len1-len2; i++) {
    // check for match @ i
    // is s2 = s1[i, i+1, ...]?
}
```