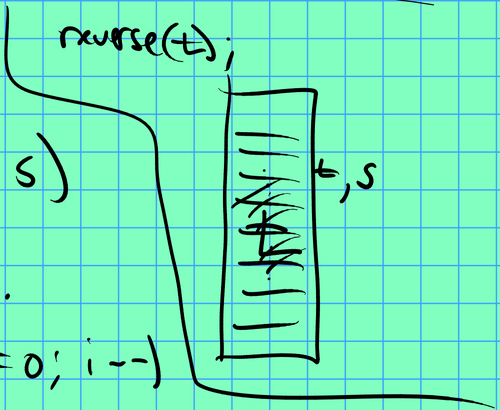
Strings : essentially just a vector<char> with a few renamed functions.

vector	string
v.push_back(x)	s += x;
v.size()	s.length()
v[i]	s[i]
	⋮

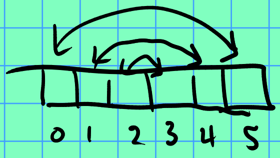
Warm up: reverse a string.

```
string reverse(const string& s)
{
        string r; // return value.
    for(int i = s.length()-1; i>=0; i--)
        r += s[i];
    return r;
}
```



what about reversing "in-place"?

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



$0 \longleftrightarrow 5$

$1 \longleftrightarrow 4$

$2 \longleftrightarrow 3$

$i \longleftrightarrow l-i-1$

$(l = $ s.length()$)$

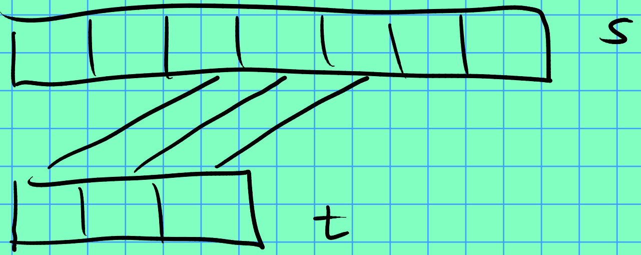
}

string ≡ vector<char>

∴ s[i] has type char.

More challenging exercise: search for a substring in another string.

```
int substr (const string& s, const string& t)
{
    // if t is a substring of s,
    // return location of first match.
    // e.g. s = "hello world.", t = "lo"
    // substr(s, t) should give 3.
    // substr(s, "nonsense") should give -1.
```



idea: check for a match starting at all possible offsets.

t[0] vs. s[i]
t[1] vs. s[i+1]
⋮

for i = 0 to s.length() - t.length()