A Function to Check Whether a `stack_t` is Full

```
// Returns 1 if stack is full, or
// 0 if stack is not full.

int32_t stack_full
    (const struct stack_t* s)
{
    return (0 == s->top);
}
```


Information Hiding and Performance Sometimes at Odds

How do we push a string without exposing details of the implementation?

For example,
- should we make a copy of the string, or
- just copy the pointer passed in?

Caller or callee
- must ensure that string does not disappear after it is pushed,
- but which one? Copying twice is wasteful.

Let's retain our current design, so **`stack_push` must make a copy**.

How Can We Handle Long Strings? Fail…

What should happen if caller passes a string longer than 199 characters?
- Fail? A valid choice, but not so useful.
- Copy the first 199? Also valid, but may not be what the user wants.
- We have no other choice with the current implementation!

We will go with failure for simplicity.

A Function to Push a String Onto a `stack_t`

```
// Returns 1 on success,
// or 0 on failure.
int32_t stack_push (struct stack_t* s,
                    const char* str)
{
    int32_t i;
    char*   write;
    if (stack_full (s)) {
        return 0;
    }
```

the stack

the string

for copying string

No space on stack? Fail.

