C-String Assignment

Assignment means "copy the thing on the right into the storage on the left". Instead of the assignment operator, used by the built-in types, C-strings use the `strcpy()` function, from the standard library header `<cstring>`, as shown below:

```
const size_t kMaxLen = 4096;
char dest [kMaxLen];
// Assume src is a C-style string
strcpy(dest, src);
```

Both `src` and `dest` are C-strings. (`src` is a common abreviation for *source*, where the characters are copied ***from***, while `dest` stands for *destination*, where the characters are copied ***to***). `strcpy(dest, src)` copies the characters, one by one, from `src` into `dest`, stopping the `'\0'` is copied. However:

- You don't know if the **actual size** of the C-string source is less than `4095` characters (+1 for the null character). **Thus this code contains a security flaw**.
- You normally won't need anywhere near `4096` characters allocated for destination, so **the code is inefficient**.

It is up to you to ensure that there is enough space in `dest` to hold a copy of `src`. The icon used in front of the code does not mean that the code is buggy; instead, it means that the **function itself is intrinsically dangerous**; it's like the symbol found on rat poison.

The library function itself **makes no attempt to check whether the destination has enough room** to hold a copy of the source string. Even if there is not enough memory the function keeps copying, possibly overwriting other data; this called a **buffer overflow**.