Instructions Follow instructions *carefully*, failure to do so may result in points being deducted. Hand in all your source code files through webhandin and make sure your programs compile and run by using the webgrader interface. You can grade yourself and re-handin as many times as you wish up until the due date.

Programs

1. In this exercise you will write several functions that operate on *strings* (arrays of `char` types). Place all function prototypes with documentation in a header file named `string_utils.h` with their definitions in a source file named `string_utils.c`. You should test your functions by writing test cases and a driver program, but you need not hand it in.

   - Write a function to determine if a given string `str` contains a substring `subStr`. The function should return true if `subStr` appears anywhere inside `str`, false otherwise.
     `int strContains(const char *str, const char *subStr);`
   - Write a function that takes two strings and concatenates them together into a new string.
     `char *concatenate(const char *s, const char *t);`
   - Write a function that takes two strings, `s` and `t` and *appends* `t` to `s` returning the result as a new string.
     `char *append(const char *s, const char *t);`
   - Write a function that takes two strings, `s` and `t` and *prepends* `t` to `s` returning the result as a new string.
     `char *prepend(const char *s, const char *t);`
   - Write a function that takes a string and returns a *new* copy of a substring of it starting from a specified index to the end of the string. For example, if we pass in `("Nebraska", 3)` it should return a new string containing `"raska"`.
     `char *substringToEnd(const char *s, int beginningIndex);`
   - Write a function that takes a string and returns a *new* copy of a substring of it starting from a specified index (inclusive) and going a specified ending index (exclusive). For example, if we pass in `("Cornhuskers", 4, 10)` it should return a new string containing `"husker"`.
     `char *substringIndex(const char *s, int beginIndex, int endIndex);`
   - Write a function that takes a string and returns a *new* copy of a substring of it starting from a specified index of a particular size. For example, if we pass in `("Lincoln", 0, 4)` it should return a new string containing `"Linc"`.
     `char *substringSize(const char *s, int beginningIndex, int size);`

2. HTML (Hypertext Markup Language) is the primary document description language (DDL) used on the web. Certain characters are not rendered in browsers as they are special characters used in HTML. For example, *tags* begin and end

with the `<, >` characters (such as `<body>`). To display such characters correctly they need to be *escaped* (similar to how you need to escape tabs `\t` and endline `\n` characters in C). Properly escaping these characters is not only important for proper rendering, but there are also security issues involved (Cross-site Scripting Attacks). [1]

You will write a program that open an HTML file, scrubs certain characters and write the results to a new output file. While there are many characters that can/should be converted to escaped-HTML characters, your function will only need to support a few. In particular, make sure that your function replaces the characters in Table 1.

HTML Character	Replacement Sequence
`&`	`&amp;`
`<`	`&lt;`
`>`	`&gt;`
`"`	`&quot;`

Table 1: Replacement Sequences

Place your source code in a file named `htmlScrubber.c`. The path/name of both the input and output files should be read in as command line arguments.

[1] In fact, many more characters need to be escaped than in this exercise to properly sanitize HTML, see https://wonko.com/post/html-escaping