

CS 211 Lab 3

Strings

Winter 2022

Today we are going to practice manipulating *strings*.

Getting Started

The starter code is available at `~cs211/lab/lab03.tgz`, so you can extract it into your current working directory with the command

```
% tar -xkvf ~cs211/lab/lab03.tgz
```

Your current working directory should now contain a subdirectory called `lab03`.

Writing the code

Navigate to your `lab03` directory and open up `src/lab3.c` in Micro:

```
% micro src/lab3.c
```

Notice that there is already some skeletons of functions and some code in `main()` here.

Now open `src/lab3_funs.c`. Note that comments explaining functionality are placed in `src/lab3_funs.h`.

```
const char* str_chr(const char* s, char c)
```

First, find the function called `str_chr`. We are going to use this function to determine if the character `c` exists in the string `s`, and if so, where. If you remember from class, we have a few ways of iterating, most notably `while` which is what you will use for this function.

Notice that `str_chr` is going to return a `const char*`.

While loops

As we learned in class, a `while` loop has the following syntax:

```
while (<expr>) {  
    // Looping through code here  
    // Until <expr> is false  
}
```

Note that in while loops we usually will use a boolean expression for `<expr>` (an expression which returns `true` or `false`.)

Use a `while` loop inside our `str_chr` in order to see if `c` is ever equal to any one of the characters in `s`. Make sure to use a `return` statement to return the `char*` if we find it (or a `NULL` if nothing is found). The

Remember that we have the `++` operator to help us.

returned `char*` should point to the first instance of the character in the string.

Once you think that your function works as intended, save and try compiling and running it. If you remember from last week, we used the *make* command in order to turn our C file into machine code. Run:

```
% make lab3
```

If everything works, if we list the files in `.`, we should now see a file called `lab3`. Enter the command

```
% ./lab3
```

See if your value looks right! If it doesn't, don't worry, Rome wasn't built in a day. Try and see what went wrong. Play around with the value of `s` and `c` to see how it affects the result.

```
bool is_prefix_of(const char* haystack, const char* needle)
```

Once we have everything working with our *str_chr*, let's move on to a similar function called *is_prefix_of*. This function is similar to *str_chr* in that it loops through a string to find something, but the difference here is that we are looking for a substring - not just a character. Also, the substring needs to be positioned at the start of the test string. Since both of the inputs are "strings" (`char*`), you will need to check that not only one character matches in the substring (*needle*), but that every character matches. Return true if the first characters of *haystack* entirely match *needle*.

Once you are done, make and run your file. See if your function properly identifies prefixes. If not, no worries, go back and try again!

```
const char* str_str(const char* haystack, const char* needle)
```

Once the function *is_prefix_of* is working, write a new function *str_str* that uses *is_prefix_of* to determine if a word exists anywhere in another word. To check if the search word (*needle*) is in the *haystack*, first check to see if it is a prefix of *haystack*. If *needle* is not the prefix to *haystack*, try to see if *needle* is the prefix of everything but the first letter of *haystack*. This loop will effectively check for the subword *needle* in every possible position inside *haystack*. Return a pointer to the start of the first instance of *needle* in *haystack* if you find it, and `NULL` if you don't.

Make and run `lab3`, and see if *str_str* works the way that you intended. Hopefully everything works! If not, as usual, go back and try and find what went wrong and update your code.

Remember, *make* works as follows:

```
% make [target]. Target is usually the name of the executable file that will be built by the make command.
```

Error messages may look scary, but in reality, they're there to help you! Not intimidate you!

Notice that *is_prefix_of* is going to return a `bool`.