

**Computer Science 215**  
**Tools and Techniques for Software Development**  
**Summer 2002**  
**Project 2 – Pointers and Strings**

**Due: Friday, 5/31/2002 1:00 p.m.**

Write a program called `str.c` that allows the user to manipulate strings in several ways. The program should be menu-driven as follows:

-----

Current value of string: ""

Please select one of the following options:

- 1 Set string
- 2 Test string for palindrome
- 3 Find substring in string
- 4 Remove substring from string
- 5 Quit

Option:

Upon start-up and after an action is performed, the program should print a dashed line (for user readability) and the current value of the string (initially empty), and allow the user to enter options until Quit is selected. Numerical error checking for the menu item is required (i.e., make sure the value is in range). Information for each of the options follows.

**1 Set string** – When this option is selected, the user is prompted for a string value (less than 30 characters). This string will replace any previous string that was entered and may contain any combination of upper/lower case characters, punctuation, spaces, etc.

**2 Test string for palindrome** – This option should print whether or not the current string is a palindrome. All punctuation and white space should be removed and all characters converted to the same case before testing.

**3 Find substring in string** – This function should print whether or not an entered substring is found in the current string. The search is case-sensitive.

**4 Remove substring from string** – This function should remove an entered substring from the current string if it is found; otherwise the current string should remain unchanged. This function is also case-sensitive.

**5 Quit** – The program should terminate when this option is selected.

For this assignment, you may **not** use any of the `str` functions (`strlen()`, `strcpy()`, etc.).

You **must** define functions with the **exact** prototypes listed below (these will *help* you):

```
void get_string (char s []);
int palindrome (char s []);
void filter_string (char s [], char n []);
int find_string (char *s, char *sub, char **loc);
int remove_string (char *s, char *sub);
```

`get_string (char s [])` should prompt the user for a string and return it in `s`. The string cannot be read in with `scanf()` since `scanf()` will terminate on any white space. Instead, use `gets()` (immediately precede it with a call to `getchar()` if you have input problems). This function should be called for menu options 1, 3, and 4.

`palindrome (char s [])` returns true (1) if `s` is a palindrome, and false (0) otherwise. This function should call `filter_string (char s [], char n [])` which should remove all the punctuation and white space from `s`, convert all the characters to upper or lower case, and return the result in `n`.

`find_string (char *s, char *sub, char **loc)` returns true (1) if `sub` is found in `s`, and false (0) otherwise. `loc` is a pointer to the position in `s` where the first occurrence of `sub` begins (if found) and `NULL` otherwise. For example if `s` is ABCD and `sub` is BC, this function should return 1 with `loc` pointing to B in `s`. Note that the empty string is a substring of every string.

`remove_string (char *s, char *sub)` returns true (1) if `sub` is found and the first occurrence removed in `s`, and false (0) otherwise. For example if `s` is ABCD and `sub` is BC, this function should return 1 and `s` should be changed to AD. If `sub` is BD, this function should return 0 and `s` should be unchanged. Note that this function should call `find_string()`.

For the first three functions above, you will manipulate `s` as an array. For the last two functions above, you will use a pointer to access `s`. I would suggest that you declare `s` and `sub` as arrays and set up your main menu handling in `main()`.

You may also find the following functions useful:

```
void print_menu ();
void print_string (char *s);
int get_menu_option ();
```

`print_menu ()` should print out the menu options.

`print_string (char *s)` should print the dashed line, the label for the current value of the string, and the string `s`.

`get_menu_option ()` should prompt the user for the menu option, check it for validity, and return a valid option; may need to call `print_menu ()`.

A sample run is available off the class web page. Additional examples may be added as needed.