Lab 7

Thursday, November 2, 2023

You will get 1 point for attending the lab, making an effort to work on the problems, and **discussing your work with a lab instructor during the lab**.

If you finish during the lab, demonstrate your working code to a lab instructor and they will give you the full 2 points. If you do not finish during the lab, discuss what you've done with a lab instructor and they will give you an opportunity to finish outside of the lab and attend office hours to demonstrate your solution.

Practice Problems. In this lab you will write a function to sort the characters of a string. First, review selection sort and trace its behaviour when applied to sort the characters in the string `"ethics"` in ascending order by ASCII value. Complete an implementation of the following functions for swapping two characters in a string:

```
// swap_to_front(str, c) swaps the initial character in the string str with the
//   character pointed to by c
// requires: str is a valid string that can be modified, length(str) >= 1
//           c points to a character in str
void swap_to_front(char str[], char * c);

// swap_to_back(str, c) swaps the last character in the string str with the
//   character pointed to by c
// requires: str is a valid string that can be modified, length(str) >= 1
//           c points to a character in str
void swap_to_back(char str[], char * c);
```

What are the number of operations used in terms of n (the length of the string) for `swap_to_front` and `swap_to_back`?

Marked Problems. Complete an implementation of the following function used to select the character of minimal ASCII value in a string.

```
// select_min(str) returns a pointer to the character of minimal ASCII value
//   in the string str (and the first if there are duplicates)
// requires: str is a valid string, length(str) >= 1
char * select_min(char str[]);
```

Complete an implementation of selection sort by using `swap_to_front` and `select_min` to place each character into its proper position in ascending sorted order. Use the following prototype:

```
// str_sort(str) sorts the characters in a string in ascending order
// requires: str points to a valid string that can be modified
void str_sort(char str[]);
```

Your implementation must use $O(n^2)$ operations in total and call `swap_to_front` $O(n)$ times where n is the length of the string. Test `str_sort` and `select_min` by (using `strcmp` and `assert` as necessary) on at least five strings each (sample strings are available in the provided code on Brightspace). You can assume the characters in the strings are all lower-case letters.