Hack 8.0

Strings & String Processing
Computer Science I

Department of Computer Science & Engineering
University of Nebraska–Lincoln

Introduction

Hack session activities are small weekly programming assignments intended to get you started on full programming assignments. Collaboration is allowed and, in fact, *highly encouraged*. You may start on the activity before your hack session, but during the hack session you must either be actively working on this activity or *helping others* work on the activity. You are graded using the same rubric as assignments so documentation, style, design and correctness are all important.

Rubric

Category	Point Value
Style	2
Documentation	2
Design	5
Correctness	16
Total	**25**

Exercises

To get more practice working with strings, you will write several functions that involve operations on strings. In particular, implement the following functions with the described behavior. You *must* use the given signatures.

1. Write a function that replaces instances of a given character with a different character in a string.
   `void replaceChar(char *s, char oldChar, char newChar);`
   Which will replace any instance of the character stored in `oldChar` with the character stored in `newChar` in the string `s`.

2. Write a function that takes a string and creates a new copy of it but with instances of a given character replaced with a different character.
   `char * replaceCharCopy(const char *s, char oldChar, char newChar);`

3. Write a function that takes a string and removes all instances of a certain character from it.
   `void removeChar(char *s, char c);`
   When removing characters, all subsequent characters should be shifted down. Take care that you handle the null terminating character properly.

4. Write a function that takes a string and creates a new copy of it but with all instances of a specified character removed from it.
   `char * removeCharCopy(const char *s, char c);`
   Take care that the new copy does not waste memory.

5. Write a function that takes a string and splits it up to an *array* of strings. The split will be length-based: the function will also take an integer n and will split the given string up into strings of length n. It is possible that the last string will not be of length n. You will not need to communicate how large the resulting array is as the calling function knows the string length and n.
   `char **lengthSplit(const char *s, int n);` For example, if we pass `"Hello World, how are you?"` with $n = 3$ then it should return an array of size 9 containing the strings `"Hel"`, `"lo "`, `"Wor"`, `"ld,"`, `" ho"`, `"w a"`, `"re "`, `"you"`, `"?"`

Instructions

- Place all your function prototypes into a file named `string_utils.h` and and their definitions in a file named `string_utils.c`.
- In addition, create a main test driver program called `stringTester.c` that demonstrates at least 3 cases per function to verify their output. Hand in your tester.
- You are encouraged to collaborate any number of students before, during, and after your scheduled hack session.
- You may (in fact are encouraged) to define any additional "helper" functions that may help you.

- Include the name(s) of everyone who worked together on this activity in your source file's header.
- Turn in all of your files via webhandin, making sure that it runs and executes correctly in the webgrader. Each individual student will need to hand in their own copy and will receive their own individual grade.