CS 100
Lab 4: cstring functions

Reminder: find a *new* partner for this lab (**50% grade reduction** for working with a person ever worked before).
You are allowed to use 1 (one) computer per team (**50% grade reduction** for using more than 1 computer per team including laptops and tablets).

Grading: 15% each function, 10% attendance

In this lab you will implement your own versions of the functions found in the **cstring** library. The functionality provided by this library will be familiar from that of the String class except that cstrings alone are unable to perform operations such as concatenation, copies, comparisons, and etc. Within this lab, **write a library titled "myCString.h"** which contains these functions. You will at a minimum need to implement at least one version (not overloaded) for each of the following functions:

1. strcpy
2. strcat
3. strcmp
4. strchr
5. strstr
6. strlen

For each implementation, **you CANNOT use the functions** provided within **the cstring library** to implement your own; however, using your own implementation of one function in another, such as strlen in strcpy, is fine. **Note**: To avoid conflicts with cstring, make sure to give a different name to your own functions (e.g., my_strcpy, my_strcmp, etc.).

You will need to look up some reference to each of these to make sure they work just like the versions in the library. Although you can look into the cstring library directly for function signatures, the best method is likely using a search engine to find a reference which contains the function signatures and explains the arguments/return value. **Note**: you should be implementing the cstring versions, NOT the string versions (i.e. all of these functions take arguments which are simple, null-terminated character arrays). Your function signatures should be **exactly the same** as the standard cstring versions and you have to test it the following way: **call cstring function and then call your version of it.** You should not implement your own string class.

In addition to writing the functions, you will need to test them. Write a main.cpp file which accomplishes this. **It is not enough to implement just one test case, your test cases need to give high confidence that they work correctly.** In addition to a normal test case, such as testing strcat with two strings "Hello" and " World", you'll need to test the boundary cases. Some examples of boundary cases includes using strings such as the empty string or strings at full capacity. Your tests will be your demonstration for this lab that your function implementations are correct so make sure you have gone over all necessary possibilities.

Submission: After demoing your programs to the TA, package your code into a single .tar.gz file and submit it to iLearn.