# CS 11 Data Structures and Algorithms

## Assignment 12: Recursion

Skip to Main Content

### Assignment 12.1 [15 points]

Write a recursive function named reverseWithinBounds that has an argument that is an array of characters and two arguments that are bounds on array indices. The function should reverse the order of those entries in the array whose indices are between the two bounds (including the bounds). For example, if the array is:

a[0] == 'A' a[1] == 'B' a[2] == 'C' a[3] == 'D' a[4] == 'E'

and the bounds are 1 and 4, then after the function is run the array elements should be:

a[0] == 'A' a[1] == 'E' a[2] == 'D' a[3] == 'C' a[4] == 'B'

Embed the function in a program and test it. After you have fully debugged this function, define another function named reverseCstring that takes a single argument that is a C string and modifies the argument so that it is reversed. This function will include a call to the recursive definition you did for the first part of this project, and need not be recursive. Embed this second function in a program and test it. Turn in only this final result (with output, of course).

### Assignment 12.2 [20 points]

In this assignment you will use the MyString class that you wrote earlier in this course.

A palindrome is a string that reads the same forward and backward. Write a program that reads in any number of MyStrings of characters from the user and determines if each MyString is a palindrome. The user will terminate each MyString by pressing the return (or enter) button. The user will indicate that there are no more MyStrings by entering "quit".

Do not make any changes to the MyString class (except, if necessary, to correct errors in your MyString class). Do not use the C++ string class.

To do this you must write a recursive function named isAPalindrome that takes a single MyString argument and two arguments that are bounds on array indices. The function must examine the part of the argument between the two bounds (including the bounds) and return true if this part of the argument is a palindrome, false if it is not a palindrome. **The function must be recursive and must not call any other functions except ispunct(), isspace(), and toupper() (described below).**

Follow this sample output closely:

```
Enter a string: Able was I, ere I saw Elba
Able was I, ere I saw Elba is a palindrome.
Enter a string: peanut butter
peanut butter is not a palindrome.
Enter a string: quit
```

In determining whether a string is a palindrome, this function must consider uppercase and lowercase letters to be the same and ignore punctuation characters and whitespace characters. You should not modify the argument in any way to accomplish this. The simplest way to handle uppercase/lowercase issues is to make everything uppercase on the fly, right at the instant that you are comparing the two characters.

**Hints:**

- You will want to use three functions that are in the cctype library (i.e. you must #include cctype to use them). They are ispunct(char), a bool function which returns true if its argument is a punctuation mark and false otherwise, isspace(char), which returns true if its argument is a whitespace character and false otherwise, and toupper(char), which returns the uppercase equivalent of its argument (without changing the argument itself).

- I strongly suggest that you first write this program so that it works in the general case (i.e. assuming that all letters are uppercase and no characters are punctuation characters or whitespace characters), and then add code to handle the uppercase/lowercase and punctuation/whitespace issues.

- Make sure you try a palindrome that starts and ends with the same letter but is not a palindrome. Returning a false positive in this case is one of the most common errors that students make.

- Another common error is calling this value-returning function as if it was a void function. You can't say "isAPalindrome(str1);" on a line by itself. It doesn't do anything.

- Also, make sure that every case in your function has a return statement. Often students submit programs that worked perfectly for them, but they were just getting lucky, and the computer was returning the correct value by chance when there was no return statement.

### Assignment 12.3 [20 points]

Write a recursive function to sort an array of integers into ascending order using the following idea: the function must place the smallest element in the first position, then sort the rest of the array by a recursive call. This is a recursive version of the selection sort. (Note: You will probably want to call an auxiliary function that finds the index of the smallest item in the array. Make sure that the sorting function itself is recursive. Any auxiliary function that you use may be either recursive or iterative.) Embed your sort function in a driver program to test it. Turn in the entire program and the output.

### Submit Your Work

You will be submitting three files, a12_1.cpp, a12_2.cpp, and a12_3.cpp.

Name your source code file(s) according to the assignment number (a1_1.cpp, a4_2.cpp, etc.). Execute each program and copy/paste the output into the bottom of the corresponding source code file, making it into a comment. Use the Assignment Submission link to submit the source file(s). When you submit your assignment there will be a text field in which you can add a note to me (called a "comment", but don't confuse it with a C++ comment). In this "comments" section of the submission page let me know whether the program(s) work as required.