CSCI 1120 (Low-Level Computing), Spring 2018

Homework 5

Credit: 15 points.

1 Reading

Be sure you have read, or at least skimmed, the assigned readings for classes through 2/07.

2 Honor Code Statement

Please include with each part of the assignment the Honor Code pledge or just the word "pledged", plus one or more of the following about collaboration and help (as many as apply).[1] Text *in italics* is explanatory or something for you to fill in. For written assignments, it should go right after your name and the assignment number; for programming assignments, it should go in comments at the start of your program(s).

- This assignment is entirely my own work. *(Here, "entirely my own work" means that it's your own work except for anything you got from the assignment itself — some programming assignments include "starter code", for example — or from the course Web site. In particular, for programming assignments you can copy freely from anything on the "sample programs page".)*
- I worked with *names of other students* on this assignment.
- I got help with this assignment from *source of help — ACM tutoring, another student in the course, the instructor, etc. (Here, "help" means significant help, beyond a little assistance with tools or compiler errors.)*
- I got help from *outside source — a book other than the textbook (give title and author), a Web site (give its URL), etc.. (Here too, you only need to mention significant help — you don't need to tell me that you looked up an error message on the Web, but if you found an algorithm or a code sketch, tell me about that.)*
- I provided help to *names of students* on this assignment. *(And here too, you only need to tell me about significant help.)*

3 Programming Problems

Do the following programming problems. You will end up with at least one code file per problem. Submit your program source (and any other needed files) by sending mail to `bmassing@cs.trinity.edu` with each file as an attachment. Please use a subject line that mentions the course and the assignment (e.g., "csci 1120 hw 5" or "LL hw 5"). You can develop your programs on any system that provides the needed functionality, but I will test them on one of the department's Linux machines, so you should probably make sure they work in that environment before turning them in.

[1] Credit where credit is due: I based the wording of this list on a posting to a SIGCSE mailing list. SIGCSE is the ACM's Special Interest Group on CS Education.

1. (5 points) In CS1 you probably learned about sorting algorithms and implemented one or more of them.[2] A simple way to test such an algorithm is to generate a sequence of "random" numbers, sort them, and check that the result is in ascending order. Sample program http://www.cs.trinity.edu/~bmassing/Classes/CS1120_2018spring/SamplePrograms/Programs/sorte shows how this might be done in C (leaving out the actual sorting). For this problem your mission is just to fill in code for the `sort` function so that it actually sorts. It's completely up to you which sorting algorithm to implement, though I'm inclined to recommend that you just do one of the simple-but-slow ones (e.g., bubble sort or selection sort). If you feel ambitious, you could try quicksort or mergesort, though mergesort is apt to be more trouble since it requires a work array.

2. (10 points) Write a C program that prompts the user for a single line of text and prints whether it is a palindrome, i.e., whether it's "the same" backwards as forwards, according to the following rules:

   - Only letters and digits count; spaces, punctuation, etc., do not.
   - Case of letters is not significant ('A' and 'a' are considered the same).

   The program should also print an error message if the text supplied by the user doesn't fit into the array you use to represent the input string.

   Here are some sample executions:

```
[bmassing@dias04]$ ./a.out
enter a line of text:
abcd dcba
input 'abcd dcba'
a palindrome

[bmassing@dias04]$ ./a.out
A man, a plan, a canal -- Panama!
input 'A man, a plan, a canal -- Panama!'
a palindrome

[bmassing@dias04]$ ./a.out
enter a line of text:
abcd 12 bcda
input 'abcd 12 dcba'
not a palindrome

[bmassing@dias04]$ ./a.out
enter a line of text:
abcd 1221 dcba
input 'abcd 1221 dcba'
a palindrome
```

   Hints:

[2] If you didn't take CS1 and don't know about sorting, ask me about supplemental reading.

- You may find sample programs http://www.cs.trinity.edu/~bmassing/Classes/CS1120_2018sprin and http://www.cs.trinity.edu/~bmassing/Classes/CS1120_2018spring/SampleProgramsProgra helpful.
- You may find library functions such as `isalpha()` and `tolower` helpful.
- In Scala you might solve this problem by doing something that involves copying the string, or parts of it. I encourage you *not* to solve it that way in C: I think it's simpler and more C-idiomatic just to work with the string as is. Consider having one index or pointer that starts at the beginning of the string and moves right and another that starts at the end and moves left.