ICS 46 Lecture B Fall 2019 Project 5: From a bedpost to a despot

Due **December 6, 2019, 11:59 PM**
Note that this is well later than the syllabus originally stated.

Introduction

We've discussed heaps a few times this quarter. We're going to discuss sorting next week. And this project is shorter due to the extension on project 4, as is this intro section.

Getting Started

Before you begin work on this project, there are a couple of chores you'll need to complete on your ICS 46 VM to get it set up to proceed.

Refreshing your ICS 46 VM environment

Even if you previously downloaded your ICS 46 VM, you will probably need to refresh its environment before proceeding with this project. Log into your VM and issue the command ics46 version to see what version of the ICS 46 environment you currently have stored on your VM. Note, in particular, the timestamp; if you see a version with a timestamp older than the one listed below, you'll want to refresh your environment by running the command ics46 refresh to download the latest one before you proceed with this project.

If you're unable to get outgoing network access to work on the ICS 46 VM — something that afflicts a handful of students each quarter — then the ics46 refresh command won't work, but an alternative approach is to download the latest environment from the link below, then to upload the file on to your ICS 46 VM using SCP. (See the Project #0 write-up for more details on using SCP.) Once the file is on your VM, you can run the command **ics46b refresh_local NAME_OF_ENVIRONMENT_FILE**, replacing **NAME_OF_ENVIRONMENT_FILE** with the name of the file you uploaded; note that you'd need to be in the same directory where the file is when you run the command.

The file is linked from the "public" ICS 46 page; click this link and enjoy the amazing web design skill that put it together: https://www.ics.uci.edu/~mikes/ics46/

Creating your project directory on your ICS 46 VM

A project template has been created specifically for this project, containing a similar structure to the basic template you saw in Project #0.

Decide on a name for your project directory, then issue the command **ics46b start YOUR_CHOSEN_PROJECT_NAME project5** to create your new project directory using the project5 template. (For example, if you wanted to call your project directory proj5, you would issue the command `ics46 start proj5 project5` to create it.) Now you're ready to proceed!

Unlike projects 3 & 4, you **may not** work with a partner this time.

Reviewing related material

I encourage you to read your textbook; in the second edition, section 8.3 discusses heaps and sorting is discussed in 11.1, 11.2, and in the week 10 handout.

Requirements

You have three functions to fill in, all to be found in the file proj5.cpp. You may not use any elements of the standard library, other than accessor functions for std::string and std::vector as needed.

- bool isMinHeap(std::string s)
    - Returns true if and only if the given string, when interpreted as an array of characters, is a min heap. Use the natural < between characters.

- void doSomeSort(std::vector<std::string> & vec)
    - Sort the given vector, using the natural <
    - **DO NOT** copy code from the general internet. Of course, that applies in any part of any assignment, but it is particular here. I am not interested in seeing if you can copy/paste MergeSort from Wikipedia.
    - **DO NOT** use a "standard library" sort, such as std::sort.
    - Implement your own, ideally based off your understanding from lecture or from the reading.

- std::vector<std::string> findHeaps(std::istream & in)
    - Read from the given stream, which will contain one or more words (defined here as any characters separated by whitespace)
    - Return a vector containing which of them are minHeaps.
    - The vector you return should be sorted.
      Do this part by calling your doSomeSort().
      You probably want to call your isMinHeap() too.

Deliverables

After using the gather script in your project directory to gather up your C++ source and header files into a single **project5.tar.gz** file (as you did in previous assignments), submit that file (and only that file) to Checkmate. Refer back to Project #0 if you need instructions on how to do that.

Before you gather, be sure your program compiles using `./build` and that, after a build, you can both `./run` and `./run gtest`.

You will submit your project via Checkmate. Keep in mind that that you're responsible for submitting the version of the project that you want graded. We won't regrade a project simply because you submitted the wrong version accidentally. (It's not a bad idea to look at the contents of your tarball before submitting it; see Project #0 for instructions on how to do that.)

Can I submit after the deadline?
Yes, it is possible, subject to the late work policy for this course, which is described in the section titled Late work in the course reference and syllabus.

Grading
Your grade for this project will be entirely based on correctness: I will run some number of test cases using Google test. Different parts of this program are worth differing amounts.

That having been said, if I notice your style is egregiously bad, I reserve the right to deduct some number of points.