CpSc 2120: Algorithms and Data Structures

Instructor: Dr. Brian Dean — Fall 2017
Webpage: `http://www.cs.clemson.edu/~bcdean/` — TTh 12:30-1:45
Handout 7: Midterm Quiz (Coding Part). 25 possible points. — McAdams 119

You may use either the lab machine in front of you or your own computer for this part of the quiz; however, for grading, your code will be compiled and run on the lab machines, and no points will be awarded if compilation fails. You can access any material from this course (e.g., homework / lab solutions, videos or lecture notes from Canvas, and previous solutions *you* have written). However, you may *not* use any other resources from the web.

Code and answers for this part of the quiz are to be submitted via handin.cs.clemson.edu, just as with the labs. You may not submit any code as part of this quiz outside of the time window for your specific lab session – any attempt to do so will be regarded as cheating.

For the questions below asking you to write a function, feel welcome to write other helper functions if needed. If you want to have other files compiled along with your code, please be sure to submit them as well to handin and add a comment to the top of your program indicating any other files that should be added to the compiler command line.

You are on your honor not to discuss this part of the quiz with any other human being until 2:30pm. Good luck, and remember that better algorithmic solutions will earn you the more points!

1. (7 points). Make a copy of the code in `filter_array.cpp` in your local directory. Then fill in the function `filter_evens` and submit the resulting file. The `filter_evens` function takes a pointer to an array of length N and returns a linked list of only the even-valued elements from the original array (in the same order as they appear in the array). The original array is not to be modified. Your program will be tested with array sizes N up to 10 million.

2. (8 points). Make a copy of the code in `filter_bst.cpp` in your local directory. Then fill in the function `filter_evens` and submit the resulting file. The `filter_evens` function takes a pointer to the root of a balanced binary search tree with N nodes and returns a balanced binary search tree consisting of only the even-valued elements from the original tree. The odd-valued elements in the original tree should be removed and de-allocated in the process, and only one copy of the even elements should exist in memory after your function is complete (i.e., unlike the previous problem, you aren't making an extra copy of the even elements). Your program will be tested with sizes N up to 1 million.

3. (10 points). The input file `password_changes.txt` contains an integer N followed by N lines of input, each containing a username and a password (each is a string of letters and numbers containing no whitespace characters). These lines describe the attempts made by various users to change their passwords over time. For example, if the first line is

```
bcdean recursion
```

then this represents an attempt by user `bcdean` to change his password to `recursion`. If a later line in the file is

```
bcdean redblacktree
```

then this means `bcdean` made a subsequent attempt to change his password to `redblacktree`.

Some password attempts are *valid*, and others are *invalid*. The first password change attempt of each user is always valid. After this, an attempt is valid only if the user attempts to change his/her password to a string that differs from the password they specified on each of their three most recent previous valid password changes (this is a fairly standard practice in computer security – not allowing a user to change their password to an old password that they recently used). Stated otherwise, a password change attempt is invalid if it tries to use a password that is the same as one of the most recent three distinct passwords chosen by the same user.

Please write the following as output:

- The total number of invalid password change attempts
- A list of users who tie for having the maximum number of invalid password change attempts (each user should be printed at most once, but they can be printed in any order).

Your program will be tested on values of N up to 1 million. For the sample input provided, the correct answers are 3 and `user1`; the invalid password changes occur on the 6th, 7th, and 9th attempts (counting from 1, not 0).