CSCI 1120 (Low-Level Computing), Fall 2016

Homework 7

Credit: 20 points.

1 Reading

Be sure you have read, or at least skimmed, the assigned readings for classes through 10/26.

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.

- This assignment is entirely my own work.
- This assignment is entirely my own work, except for portions I got from the assignment itself *(some programming assignments include "starter code")* or sample programs for the course *(from which you can borrow freely — that's what they're for).*
- 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.*
- I got significant help from *outside source — a book other than the textbook (give title and author), a Web site (give its URL), etc.. ("Significant" here means more than just a little assistance with tools — 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 significant help to *names of students* on this assignment. *("Significant" here means more than just a little assistance with tools — you don't need to tell me about helping other students decipher compiler error messages, but beyond that, do tell me.)*

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 7" or "LL hw 7"). 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. (20 points) Your mission for this assignment is to complete a partial implementation in C of a binary search tree (a.k.a. sorted binary tree) of `int`s. (I'm hoping that all of you know about this data structure from CS2. If you don't — the Wikipedia article[2]s a reasonable description (but I recommend that you not read the example code until/unless you try to write your own).

   This partial implementation consists of a number of files:

   - Function declarations for tree: int-bst.h[3].
   - Starter file for function definitions: int-bst.c[4].
   - Test program and supporting files: test-int-bst.c[5], test-helper.c[6], test-helper.h[7].
   - Makefile for compiling (comments in the file tell you how to use it): Makefile[8]. *NOTE* that you should use your browser's "download" or "save" function to obtain this file, rather than copying and pasting the text. This is because copy-and-paste will likely replace the tab characters in the file with spaces, with bad consequences (since tabs are semantically significant in makefiles.)

   Your job is to modify the file `int-bst.c` so it includes function definitions for all the functions declared in `int-bst.h`. (You may want to add additional "helper" functions, but if so they should probably go only in `int-bst.c`.) Notice that the function that removes a single element of the tree (`int_bst_remove`) is optional — you can provide an "implementation" that just prints an error message, or for extra credit you can actually implement this operation. You should not modify any other files, unless you want to add additional tests to `test-int-bst.c`.

   Sample output of the test program:

```
inserting 40 into tree [ ]
result [ 40 ]
inserting 30 into tree [ 40 ]
result [ 30 40 ]
inserting 50 into tree [ 30 40 ]
result [ 30 40 50 ]
inserting 20 into tree [ 30 40 50 ]
result [ 20 30 40 50 ]
inserting 60 into tree [ 20 30 40 50 ]
result [ 20 30 40 50 60 ]
inserting 16 into tree [ 20 30 40 50 60 ]
result [ 16 20 30 40 50 60 ]
inserting 14 into tree [ 16 20 30 40 50 60 ]
result [ 14 16 20 30 40 50 60 ]
inserting 18 into tree [ 14 16 20 30 40 50 60 ]
```

[2] i

[3] http://www.cs.trinity.edu/~bmassing/Classes/CS1120_2016fall/Homeworks/HW07/Problems/int-bst.h

[4] http://www.cs.trinity.edu/~bmassing/Classes/CS1120_2016fall/Homeworks/HW07/Problems/int-bst.c

[5] http://www.cs.trinity.edu/~bmassing/Classes/CS1120_2016fall/Homeworks/HW07/Problems/test-int-bst.c

[6] http://www.cs.trinity.edu/~bmassing/Classes/CS1120_2016fall/Homeworks/HW07/Problems/test-helper.c

[7] http://www.cs.trinity.edu/~bmassing/Classes/CS1120_2016fall/Homeworks/HW07/Problems/test-helper.h

[8] http://www.cs.trinity.edu/~bmassing/Classes/CS1120_2016fall/Homeworks/HW07/Problems/Makefile

```
result [ 14 16 18 20 30 40 50 60 ]
inserting 24 into tree [ 14 16 18 20 30 40 50 60 ]
result [ 14 16 18 20 24 30 40 50 60 ]
inserting 56 into tree [ 14 16 18 20 24 30 40 50 60 ]
result [ 14 16 18 20 24 30 40 50 56 60 ]
inserting 64 into tree [ 14 16 18 20 24 30 40 50 56 60 ]
result [ 14 16 18 20 24 30 40 50 56 60 64 ]
inserting 30 into tree [ 14 16 18 20 24 30 40 50 56 60 64 ]
result [ 14 16 18 20 24 30 40 50 56 60 64 ]
inserting 50 into tree [ 14 16 18 20 24 30 40 50 56 60 64 ]
result [ 14 16 18 20 24 30 40 50 56 60 64 ]
test data in order [ 14 16 18 20 24 30 30 40 50 50 56 60 64 ]
40
  30
      20
          16
              14
                  .
                  .
              18
                  .
                  .
          24
              .
              .
      .
  50
      .
      60
          56
              .
              .
          64
              .
              .
finding 0 in tree [ 14 16 18 20 24 30 40 50 56 60 64 ]
result false
finding 100 in tree [ 14 16 18 20 24 30 40 50 56 60 64 ]
result false
finding 10 in tree [ 14 16 18 20 24 30 40 50 56 60 64 ]
result false
finding 40 in tree [ 14 16 18 20 24 30 40 50 56 60 64 ]
result true
finding 14 in tree [ 14 16 18 20 24 30 40 50 56 60 64 ]
result true
finding 64 in tree [ 14 16 18 20 24 30 40 50 56 60 64 ]
result true
removing 0 from tree [ 14 16 18 20 24 30 40 50 56 60 64 ]
```

```
result [ 14 16 18 20 24 30 40 50 56 60 64 ]
removing 16 from tree [ 14 16 18 20 24 30 40 50 56 60 64 ]
result [ 14 18 20 24 30 40 50 56 60 64 ]
removing 60 from tree [ 14 18 20 24 30 40 50 56 60 64 ]
result [ 14 18 20 24 30 40 50 56 64 ]
removing 30 from tree [ 14 18 20 24 30 40 50 56 64 ]
result [ 14 18 20 24 40 50 56 64 ]
removing 50 from tree [ 14 18 20 24 40 50 56 64 ]
result [ 14 18 20 24 40 56 64 ]
40
  20
      14
          .
          18
              .
              .
      24
          .
          .
  64
      56
          .
          .
      .
inserting 0 into tree [ 14 18 20 24 40 56 64 ]
result [ 0 14 18 20 24 40 56 64 ]
inserting 100 into tree [ 0 14 18 20 24 40 56 64 ]
result [ 0 14 18 20 24 40 56 64 100 ]
inserting 0 into tree [ 0 14 18 20 24 40 56 64 100 ]
result [ 0 14 18 20 24 40 56 64 100 ]
inserting 100 into tree [ 0 14 18 20 24 40 56 64 100 ]
result [ 0 14 18 20 24 40 56 64 100 ]
after removing all elements [ ]
```

Hint: You may find it helpful to look more closely at the sorted-list example shown in class and available on the course "sample programs" page — it's meant to be a model for one way to implement a linked data structure in C, and the functions you need to write code for are meant to be tree versions of functions in `sorted-int-list.c`. It's up to you whether to use recursion or iteration or both, but I advise that recursion will probably be much easier for the two functions that print the tree.

What to turn in: Just send me your `int-bst.c` file, unless you added more tests to `test-int-bst.c`, in which case send that too (but *be sure* your code works with the provided version as well).