CECS 460 - Lab 1

Created by: Haixia Peng

1. Introduction

This lab has been designed to help you recall your C coding capability. The C language will be used in the other labs in this course. You are expected to complete all exercises within the first three weeks and prepare a lab report for submission.

2. About the Source Code

The Lab 1 source code that has been provided to you implements a binary search tree (an example shown in Fig. 1). You may wish to search online or consult an algorithms textbook to review the fundamentals of this data structure. The following table provides a list of the files contained in CECS460_Lab1_Source_Code.zip.

File Name	Description
bst.c	The binary tree data functionality (class) is implemented in this file.
bst.h	The associated header file for bst.c. Contains the function prototypes and defines the bsn_t and bst_t structs.
p1_main.c	The main function is implemented here. This file also contains code to test the binary tree data structure. The code performs a series of insertions followed by a series of deletions to verify the implementation.
type.h	Contains definitions for the types used in this lab.

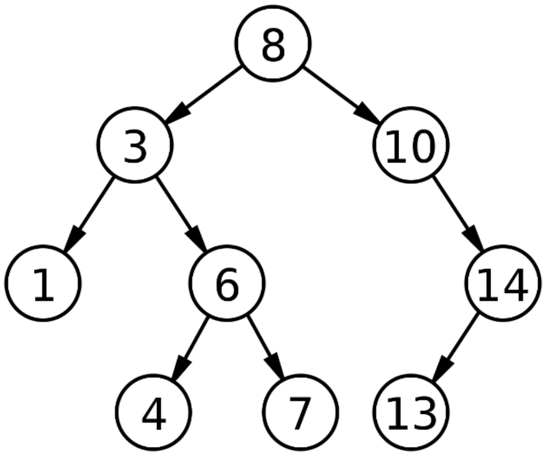

Fig.1 An example of binary search tree

The binary search tree struct stores the address of the root node and the number of nodes currently stored in the tree. This is implemented using the bst_t structure, as defined in bst.h. Every node within the tree stores the addresses of both the left and right children (NULL if there is no child there) as well as the integer value stored in this node (of type S32). This is implemented using the bst_n structure, as defined in bst.h. The bst.c file implements the following functions (with some errors you will need to fix) to manipulate the binary search tree with n nodes and a height h:

1. void bst_init(bst_t *);

   Initialize the binary search tree so that it is empty. Run time: Θ(1)
2. U32 bst_size();

   Return the number of nodes in the binary search tree. Run time: Θ(1)
3. bool bst_insert(bst_t *, S32);

   Insert the given integer into the binary search tree and return false if the node is already in the tree (do not add a duplicate into the tree) and true otherwise. Run time: O(h)
4. S32 bst_min(bst_t *);

   Returns the smallest integer in the binary search tree. Return INT_MAX if the tree is empty. Run time: O(h)
5. S32 bst_max(bst_t *);

   Returns the largest integer in the binary search tree. Return INT_MIN if the tree is empty. Run time: O(h)
6. bool bst_erase(bst_t *, S32);

   If the object is in the binary search tree, remove it and return true; otherwise, return false and do nothing. Run time: O(h)
7. #include <stdbool.h> has been added to allow access to the type bool.
8. #include <limits.h> is used to access INT_MIN and INT_MAX.

3. What You Need to Do

1. **Fix the syntax errors**: Using any of your existing compiler or makefile to compile the provided codes. In case you need it, a document called "A Simple Makefile Tutorial" has been provided. When you compiling the source code, you should have around 30 errors. Read the errors and correct the problems that are causing them.

   (**Note,** when preparing your lab report, please include a section called "Syntax Errors Fixing". In this section, please give out the line number in which each error was and show how did you fix it. Moreover, please indicate the compiler that you used for this lab.)
2. **C Formatting and Style**: Try your best to improve the formatting and style. You may make changes to main.c or bst.c. (**Note**, in your report, please include a section called "C Formatting and Style". In this section, you should record at least 5 changes you made to the code to improve the formatting and style. Include the function and line number in which the change was made and a brief justification for the change.)
3. **Logic Bugs**: Code that runs is the first step. You also need to make sure it runs properly. Using your debugging techniques to help you fix the logic bugs in the provided code. Once all the bugs are fixed, the printed output results should be the same as the following table. (**Hint**: All the comments given in the code are correct. There are 5 logic bugs in total and all logic bugs are in bst.c file)

	Min	Max
Before first group erased:	-9	9593
After first group erased:	-9	9593
After second group erased:	13	9593
After third group erased:	140	9593

After fourth group erased:	140	9265
After fifth group erased:	2147483647	-2147483648

(**Note**, in your report, please include a section called "Logic Bugs Fixing". In which, you should record all the logic bugs that you found in the provided code. Include the function where the bugs were found and a brief description of how you find and fix these bugs. Also, please put a screenshot that shows the output results in this section.)

4. Marking

To receive credit, you will need to do the following:

1. Upload your bst.c and p1_main.c files to BeachBoard.
2. Prepare and upload your report (at least two pages) to BeachBoard. Please ensure to include the three above-mentioned sections in your report. Also, for the students who finished the lab in a group, please give out a brief description of each member's responsibility in the lab design.

5. Submission Deadline

Feb. 7, 2022, 11:00 pm