Project: Binary Search Trees

COSC 216.001

Due Date: Oct 14, 2021, 11:59pm

1 Overview

Each student is asked to implement an API for Binary Search Trees in C.

1.1 Objectives

- Use C dynamic memory management functions - `malloc` & `free`
- Implement a data structure using `struct`s and pointers
- Additional practice with recursion

2 Detailed Description

2.1 Introduction

In Computer Science, a binary tree is a hierarchical structure of nodes, each node referencing at most to two child nodes. Every binary tree has a root from which the first two child nodes originate. If a node has no children, then such nodes are usually termed leaves, and mark the extent of the tree structure.

A particular kind of binary tree, called the binary search tree, is very useful for storing data for rapid access, storage, and deletion. Data in a binary search tree are stored in tree nodes, and must have associated with them an ordinal value or key; these keys are used to structure the tree such that the value of a left child node is less than that of the parent node, and the value of a right child node is greater than that of the parent node. Sometimes, the key and datum are one and the same. Typical key values include simple integers or strings, the actual data for the key will depend on the application.[1]

Note: this project does not call for *balanced* binary search trees. Although, you might consider what additions and changes would be necessary to achieved balanced trees in the future.

[1]taken from: `http://www.codeproject.com/KB/recipes/BinarySearchTree.aspx`

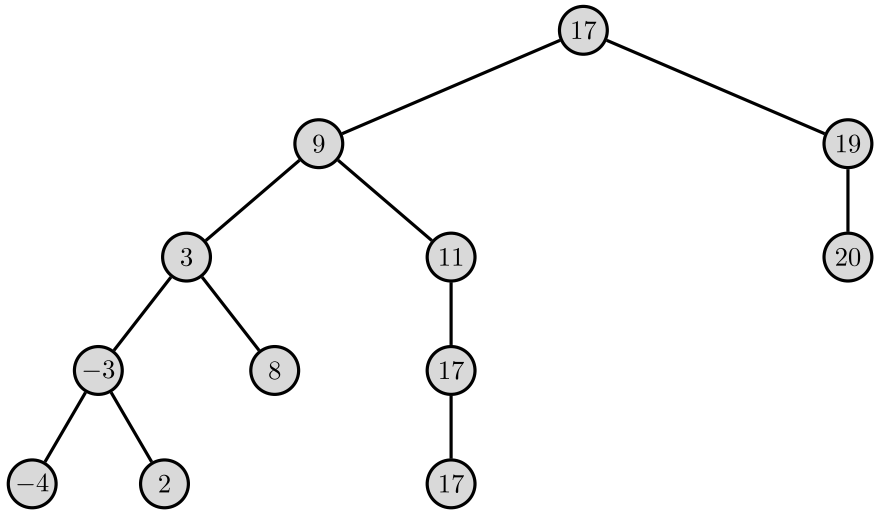

Figure 1: an example of a binary search tree

2.2 What to do

Create a C file called "bst.c" that implements binary search trees of strings. Your implementation should make no assumptions about the number of nodes, or the eventual structure of the BST. Therefore, you should use C dynamic memory allocation (`malloc()`) to create space for each node as necessary. You should also clean-up nodes that are removed or destroyed using `free()`. The header file "bst.h" is available on Blackboard; your project should implement all of the functions declared in the header file. Specifically, you will be implementing:

```
int bst_create  ( bst *newTree );
int bst_insert  ( bst *theTree, char *value );
int bst_first   ( bst *theTree, char *dst );
int bst_next    ( bst *theTree, char *dst );
int bst_previous( bst *theTree, char *dst );
int bst_last    ( bst *theTree, char *dst );
int bst_find    ( bst *theTree, char *value );
int bst_remove  ( bst *theTree, char *value );
int bst_destroy ( bst *theTree );
```

NOTE: You should not modify the provided header file without our permission

Your file will not run on it's own, but will need to be called from some other source file that contains a main(). For testing purposes, you might implement a small program (called a **driver** program) that exercises your binary search tree implementation.

2.3 Compiling you program

Please use gcc to compile and submit your program. specifically use the following command to compile your program:

```
gcc -Wall -pedantic-errors -Werror -c bst.c
gcc -Wall -pedantic-errors -Werror <driver.c> bst.o -o test_driver
```

Replace <*driver.c*> with the filename for your driver program. I chose `driver.c` for mine, and I suggest you do the same. We'll explain the other options in class, but the result should be a program called `test-driver` All your C programs in this course should be written in correct C, which means they must compile and run correctly when compiled with the compiler `gcc`, with the options `-Wall, -pedantic-errors, and -Werror`. Except as noted below, you may use any C language features in your project that have been covered in class, or that are in the chapters covered so far and during the time this project is assigned, so long as your program works successfully using the compiler options mentioned above.

3 Submission

Your source code file must have a comment near the top that contains your name, StudentID, and M-number.

Project should be submitted via Blackboard by **October 14, 2021, 11:59pm**. Follow these instructions to turn in your project.

You should submit the following files:

- Makefile
- bst.c
- *any other source files your project needs*

The following submission directions use the command-line submit program that we will use for all projects this semester.

1. create a directory for your project:

   ```
   mkdir bst
   ```

2. create (or copy) all of your source files in this directory. Example: To copy a file called `example.c` into your `bst` directory:

   ```
   cp example.c bst
   ```

3. change parent directory of your project directory:

   ```
   cd bst/..
   ```

4. Create a tar file named `<user_name>.tar.gz`, where `<user_name>` is your studentID and `<proj_dir>` is the directory containing the code for your project, by typing:

   ```
   tar -czf <user_name>.tar bst
   ```

5. Finally, submit the compressed tar file to Blackboard in accordance with the class policies.

Late assignments will not be given credit.

4 Grading

While this rubric is subject to change based on class performance, the current grading rubric for this assignment is as follows:

component	value
Correctness	40
Completeness	40
Code Quality	20