Arizona State University
CSE 310

Instructor - Nakul Gopalan
TA - Weiwei Gu, Alma Babbit, Anushka Tiwari
Graders - Suresh Kondepudi, Lakshya Jain, Adarsh Hiremath

Project 4: Binary Search Tree

Important Dates:
Homework Released: Oct. 24, 0:00 am
Homework Due: Nov. 4, 11:59 pm

ASU Academic Integrity:
Students in this class must adhere to ASU's academic integrity policy, which can be found at https://provost.asu.edu/academic-integrity/policy. If you are caught cheating you will be reported to the Fulton Schools of Engineering Academic Integrity Office (AIO). The AIO maintains a record of all violations and has access to academic integrity violations committed in all other ASU colleges/schools.

What to submit:
You need to submit your code with the following files:

1. Makefile: To build the executable file
2. main.cpp: Main function

You need to submit these files to the "Project 4" assignment on Gradescope.

Introduction:
In this assignment, you will implement a fully functional binary search tree with the appropriate tree modifying and traversal methods. You will code the Insert, Delete, Search, InOrder, PreOrder, PostOrder, LevelOrder methods. You will use these to process a series of user commands to modify and traverse the binary search tree. You can use any number of helper methods to assist in coding the given methods you will be graded on. You may also use any pre-existing library for the queue data structure for the LevelOrder method.

Inputs:
The inputs consist of an integer number_of_queries (1 <= number_of_queries <= 10000) that describes the number of queries to process, followed by the queries, where the number_of_queries comes as the first line of the inputs, and queries are separated into different lines. Each of these queries is from one of the followings:

- INSERT value(e.g., “INSERT 10”): Insert an integer value into the binary search tree(1<= value <= 1000).
- DELETE value(e.g., “DELETE -32”): Delete an integer value (if it exists) from the binary search tree (1 ≤ value ≤ 1000).
- SEARCH value (e.g., “SEARCH 785): Search for an integer value in the binary search tree (1 ≤ value ≤ 1000).
- INORDER: Print the binary search tree in order traversal.
- PREORDER: Print the binary search tree preorder traversal.
- POSTORDER: Print the binary search tree postorder traversal.
- LEVELORDER: Print the entire binary search tree in level order traversal starting from level 0 to all the levels.

Outputs:

- For each “INORDER”, “PREORDER”, “POSTORDER”, “LEVELORDER” command, display the appropriate tree traversal in one line separated by a space for each element(e.g.,16 54 93). **There must be a space after each number, even if there is only one node being traversed.**
- If the binary search tree is empty when one of the traversal methods are called, print an empty line.
- For the “SEARCH” command, print out ‘True’ if the element exists in the binary search tree and ‘False’ otherwise on their own separate line.

Examples:

We provide two examples to illustrate the behavior of this data structure upon different queries.

Example 1:
Inputs:

```
4
INSERT 30
INSERT 15
DELETE 15
INORDER
```

Outputs:

```
30
```

Explanation:

- The first command inserts 30 into the binary search tree.
- The second command inserts 15 in the binary search tree.
- The third command deletes 15 from the binary search tree.
- The fourth command prints the in order traversal of the binary search tree, which only contains a single node with value 30.

Example 2:

Inputs:

```
10
INSERT 96
INORDER
DELETE 420
INSERT 420
POSTORDER
SEARCH 96
DELETE 420
DELETE 96
LEVELORDER
SEARCH 351
```

Outputs:

```
96
420 96
True

False
```

Explanation:

- The first command inserts 96 into the binary search tree.
- The second command prints the in order traversal of the binary search tree. (Notice that there is a space after 96)
- The third command prints any empty line since the given element does exist in the binary search tree.
- The fourth command inserts 420 into the binary search tree.
- The fifth command prints the postorder traversal of the binary search tree.
- The sixth command prints on a new line True since 96 exists in the binary search tree
- The seventh command deletes the existing element in the binary search tree
- The eighth command deletes the existing element in the binary search tree
- The ninth command prints an empty line since there are no elements in the binary search tree to traverse
- The tenth command searches for a non-existent element in the binary search tree and prints

False

Submission Guidelines:

1. You will need to submit two files:
    a. main.cpp
    b. Makefile

    No skeleton code is provided for this project; you need to write all these files on your own.
2. Please name the executable of the project “MAIN” in your Makefile as we will run the executable file with the command of “***./MAIN***”. Notice that this is case sensitive.
3. Save your changes and commit frequently to the Github repository as it is good coding practice.
4. You will need to submit your code on **Gradescope**. The procedure of submission is as follows:
    a. Go to Gradescope on Canvas page of the CSE 310 course
    b. Open “Project 4” in gradescope and submit your “main.cpp” and “Makefile”. Observe the failed test cases. You can correct your code and submit it as many times as you want.