# California State University, Fresno

# DEPARTMENT OF COMPUTER SCIENCE

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Class: | **Algorithms & Data Structures** | | | Semester: | **Fall 2021** |
|  | | | | | |
| Points |  | Document author: | **Marlon Branham** | | |
|  | Author’s email: | **zekebranham@mail.fresnostate.edu** | | |
| Laboratory number: | **Laboratory 2** | | |
|  | | | | | |

**1. Statement of Objectives**

In this lab, we were to develop a program that would implement a Binary Search Tree. We are to use multiple functions to explore the fundamentals of BSTs. The purpose of this lab is to understand the BST data structure and how to work with different BST functions. This was accomplished through working on this program. In this report, I will be discussing my experimental procedure, analysis of results, encountered problems, and a conclusion.

**2. Experimental Procedure**

We were to create a BST that would create a tree given certain values, a function to insert items into the BST, a function to search for a value in the tree, a function to print the maximum value of the tree, and a function to print the tree in Post-Order traversal. This was accomplished by creating two classes, one named ‘BSTNode’ that provides the elements of each node, and the second class named ‘BST’ that uses a default constructor to create a root node. The second class ‘BST’ holds my functions, being ‘InsertNode/InsertNodeHelper’, ‘search/searchHelper’, ‘maximum’, ‘PrintPostOrder/PrintPostOrderHelper’. All of these functions are defined in by names, with concise code to accomplish what the names are meant to be.

In main, I inserted nodes by individually calling the ‘InsertNode’ function on each value. I also included a commented section that uses a random number generator that creates random 500,000 numbers to be inserted into the tree from a range of 1-499,999 numbers. The program is robust enough to handle this date easily with a quick runtime.

A computer screen with text and numbers

Description automatically generated

**3. Analysis**

My results replicated the given sample output document close to perfect. The results of this lab are limited since there is not much to report. I successfully implemented all of the functions with good runtime and no bugs. Of course, when using the random number generator with 500,000 elements, viewing the inserted elements and the Post-Order traversal becomes a bit difficult.

A screenshot of a computer screen

Description automatically generatedA black screen with white text

Description automatically generated

**4. Encountered Problems**

Luckily, there were no encountered problems with this lab. Everything was implemented and ran smoothly just as the description guided.

**5. Conclusions**

What I learned in this lab was that the idea behind BST data structures is straightforward. Understanding the implementation of the BST structure with both classes and structures is definitely useful in understanding the true methods of implementation. I would prefer to use a structure formatted program, but using classes is just as simple as structures. Overall, a successful lab where I learned how to work with Binary Search Trees.

**6. References**

None.