CPSC 2150 – Algorithms and Data Structure II

Lab5: Binary Trees

Total - 50 Marks

"Good design adds value faster than it adds cost."
- Thomas C. Gale

Learning Outcomes

- Design, implement and analyze the tree-based algorithms in terms of time and space complexity.
- Design and implement the operations of binary search tree and expression trees.
- Write recursive solutions to non-trivial problems, such as binary search tree traversals.
- Develop C++ code based on the existing constraints.

Resources

- Chapter 7 of the text book

Description

This is a practice on design, implement and analyzing problems using binary search trees. The main function of your test program (testBST.cpp) should be compatible with the following main function. By compatible it means the number of parameters of each function should match. The type of parameters is determined by your design and implementation. This makes some constraints in your design, which is likely to occur in the real world.

```
int main() {
  // Fill this part to declare your variables ...
  n1 = getInput(); // it is a non-negative integer that can be read from user or
                   // can be generated randomly by computer
  list1 = genData(n1); //generates a list of n1 random numbers [-n1, n1]
  cout << "The List1: ";
  printList(list1); //prints elements of the given list

  bst1 = makeBST(list1);
  printBT(bst1);

  cout << "The height of bst1 is " << height(bst1) << endl;

  return 0;
}
```

Exercise 1: BST class **(BST.h)**

[30 marks] Provide an appropriate data structure and necessary methods to build a binary search tree which enables you to complete the rest of this assignment. Define your class with the generic types.

Exercise 2: Binary Search Tree **(testBST.cpp)**

a. **[3 marks]** Write a function named **genData** that given an integer **n**, generates a list of **n** random integers in the interval **[-n, n]** (square bracket means inclusive).

b. **[3 marks]** Write a function named **makeBST** that given a list of data, generates a binary search tree.

c. **[3 marks]** Write a function named **printBT()** that given a binary tree, prints the tree's elements using in-order and pre-order traversal.

d. **[3 marks]** Write a function named **height()** that finds the height of the binary tree which has been passed as its parameter.

Exercise 3: Time Complexity (**answers.pdf**)

a. **[8 marks]** Calculate the time complexity of your functions in Exercise 2; i.e. genData(), makeBST(), printBT(), and height().

SUBMIT to D2L

Submit a zip file named **StudentNumber-Lab5.zip** including all related files such as **answers.pdf** and **testBST.cpp and BST.h** by the due date. For example, if your student number is 10023449, the submitted file must be named as **10023449- Lab5.zip**.