Assignment 4

Trees and AVL Trees

Data Structures (Section A & B), Fall 2018

The Assignment 4 consist of two parts and each part have different deadline. Late submissions will not be accepted.
Due Date Assignment 4 PART 1**: 12:00 pm, 30th October 2018**
Due Date Assignment 4 PART 2**: 12:00 pm, 6th November 2018**

Submission Location: Upload the zip file containing your solution on the Google classroom. The name of the ZIP file should be your Roll Number.

ASSOCIATIVE MAPS

Associative Maps or Tree Maps are an important data structure that have many uses in real-world. Conceptually, Map is a data structure that consists of are collection of **key and value pairs.** In a map, the *key values* are generally used to sort and uniquely identify the elements, while the *mapped values* store the content associated to this *key.*

Problem Statement

Consider an online shopping mart. The online shopping marts advertise and display million of the products on their website. However, they do not have all the product in stock. The products that are most frequently purchased by customers are kept in warehouse while the rest of the products are bought when a demand is placed by a customer.

We wish to keep track of the most frequently bought products along with their name and quantity on hand (quantity available in ours stock). So, whenever a customer placed an order we can quickly check if the product is available in our stock or not.

Conceptually, we want to have an associative map of products:

Key (product id)	**Value (product name, quantity available in stock)**
123	**Name**: AMD GPU **Quantity:**5
2098	**Name:** Intel Corei7 CPU **Quantity:**10
567	**Name:** Kingston RAM **Quantity:**350
39911	**Name:** DDR3 RAM **Quantity:**50

PART 1) Implement **Associative MAPS** using **template based Binary Search Trees (BST)** where data of the BST will be of type Product.

1. **Provide following operations in your BST class**
   - **a) Insert:** Provide **Recursive as well as Iterative** Insert function. The Insert function will insert a given value in BST. Incase value already exist then it should return false and do not insert.
   - **b) Delete:** Provide **Recursive as well as Iterative** Delete function. The delete function will delete the node with given key. Incase key does not exist then it should return false. Your code will perform deletion by copy.
   - **c) LookUp:** Provide Recursive as well as Iterative function to find if the record with the given Key exists.
   - **d) Print:** Print the records in BST, sorted by the key. You will provide two versions of the print
     - **i. Recursive**
     - **ii. Iterative using stack.**

e) **Duplicate:** A function to create a copy of the BST.

f) **Node_Count:** Give a function to count the number of nodes in the tree.

g) **T * LargerValues (T value)**: This function will return an array consisting of all the values larger than the given value in sorted order. So, in case of BST of products. It will return all the products with id greater than the given product id.

h) **Destructor**

i) **Load: create a BST** from a file data. The file will consist of ID, Name and Quantity. File format will be as follows (the fields are separated by tab)

ID	**Name**	**Quantity**
123	**AMD RAM**	**75**
245	**K RAM**	**47**

2. Explain why implementing Associative Maps using array or linked list is not a good idea.

```
template<class T>
class BSTNode {
public:
        T data;
        BSTNode<T> *left, *right;
};
```

```
template<class T>
class BST {
  /*Add all the functions here */
private:
        BSTNode<T>* root;
}
```

```
class Product {
public:
/* Provide all required operations here*/
/* Overload all the required operators needed for the
proper functioning*/
private:
        int ProductID; //Key
        string name; //Value
        int quantity; //Value
};
```

```
int main() {
    cout << "Associative Map using BST";
    BST<Product> ProductMap;
    return 0;
}
```

NOTE: Do not create an object of PRODUCT in any function of BST class.

BST should not know about the PRODUCT class.

BST will compare data of different nodes and since data is of type T compiler will call the operators in Product class.

Overload all the operators that you need in PRODUCT class.

PART 2) Implement **Associative MAPS** using **template based AVL trees** where data will be of type Product.

- **Provide following operations**
    a) **Insert**
    b) **Delete**
    c) **LookUp**
    d) **Print**
    e) **Duplicate**
    f) **Count the number of Records**
    g) **Destructor**
    h) **Load: input the tree from a file**
    i) **Find Maximum key**
    j) **T * LargerValues (T value**): This function will return an array consisting of all the values larger than the given value in sorted order. So, in case of BST of products. It will return all the products with id greater than the given product id

- **Which implementation of Associative Map is better? Maps using AVL tree or Map using BST. Why?**

CODE DESIGN GUIDLEINES

- Do template-based programming
- Code should be properly intendent and commented (2 marks for this)
- Make sure there are no memory leaks or dangling pointers
- Don't cheat or take too much unnecessary help from your friends