National University of Computer and Emerging Sciences, Lahore Campus

Course:	Data Structures Lab	Course Code:	CL218
Program:	BS(Computer Science, Software Engineering)		
Duration:	2.5 Hours	Semester:	Spring 2022
Paper Date:	29-Jun-2022	Total Marks:	100
Section:	All	Weight	40%
Exam:	Final	Page(s):	4
		Roll No:	

Instruction/Notes:

- Need to Create **TWO** .cpp files(**19I-0000_Q1.cpp, 19I-0000_Q2.cpp**)
- **1)** Create a folder.**2)** Folder name **YourName_Rollno(e.g: FrazYousaf_19I_2447). 3)** Copy and Paste your both .cpp files in your folder **4)** And just **Drag and Drop** your folder into the submission folder. **(Please don't zip the folder).**
- Submission path: **\\cactus1\xeon\Spring 2022\Fraz Yousaf\ Final Submission\ Data Structure\Your Section.**
- **Double check your both .cpp files before submission.No issues will be entertained afterwards.**
- Path for **Q1: main.cpp: \\cactus1\xeon\Spring 2022\Fraz Yousaf\ Final Submission\Data Structures\main.cpp**
- Path for **Q2: booksDataFile: \\cactus1\xeon\Spring 2022\Fraz Yousaf\ Final Submission\Data Structures**books_data
- You are not allowed to have any helping code with you.
- Understanding the problem statement is part of your exam, so you are not allowed to ask the invigilator any questions.
- Cell phones are not allowed and Plagiarism will result in F grade.

Question 1: [30]

You have been provided with a TNode struct implementation in main.cpp file:

```
template <typename k, typename v>
struct TNode {
k key;
v value;
int nodeCount;
TNode<k, v> *left, *right ;
};
```

(nodeCount in any node n_i contains the count of total number of nodes in the tree rooted at node n_i. In other words, n_i->nodeCount= n_i->left->nodeCount + n_i->right->nodeCount + 1. The nodeCount of any leaf node is 1.)

Now complete the implementation of the class **BinaryTree** (present in main.cpp) which is a binary tree containing a pointer to the root of the tree, and in which the absolute difference in the total number of nodes in the left and right subtrees of any non-leaf node can be at most one:
|total_nodes_in_left_subtree – total_nodes_in_right_subtree| <= 1 for all non-leaf nodes. We will call this property the **node count difference property**. The default constructor and destructor for BinaryTree class have already been implemented in main.cpp.

Examples of Binary Trees that satisfy the **node count difference property** are:

```
Binary Tree 1          Binary Tree 2          Binary Tree 3
     10                     13                     45
   /    \                  / \                    /
  2       0              23   15                 87
 / \     /
3   10   1
```

You have to implement the following member functions for BinaryTree class (You are not allowed to change the function signature. There must not be any **memory leaks** or **dangling pointers** in your implementation. You can make any helper member functions if you need):

1. A function named "insert" which inserts the given <key, value> pair in the tree while maintaining the **node count difference property**. Duplicate keys will be allowed in the tree. The insert method will prefer to insert the new node in the left subtree of a node n_i of the binary tree, if **node count difference property** is not violated for node n_i by inserting in its left subtree; otherwise it will insert the new node in the right subtree of node n_i. `void insert(k const key, v const value)` **(10 marks)**

2. A function named "search" which searches for the given key passed as parameter and returns the total number of times the key is present in the tree. `int search(k const key) const` **(10 marks)**

3. A function named "isPerfect" which returns true if the binary tree is a perfect binary tree. A perfect binary tree is a tree in which every non-leaf node has two children, and every leaf node is at the same level. `bool isPerfect() const` **(10 marks)**

Question 2: [70]

(There is no main.cpp file for this question you need to create yourself)

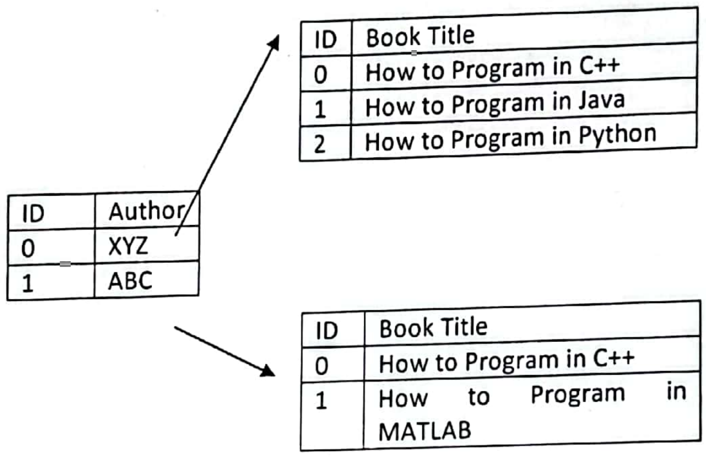
FAST-Library wants to upgrade its system for storing books. The current system is slow, as it is based on arrays. The first plan was to store the data in the form of balanced trees, but FAST management had rejected this idea as it would create difficulties while adding thousands of books. Every entry might need to balance the tree that is time consuming. The FAST development team and management team agreed on hashing technique for their new system.

You task is to build a system for storing the books in the hash tables.

have two types of specifiers for books. The first is author name, while the second is book title. The system will e based on the hash of hash tables. The first hash table contains the author's name. All the books written by a particular author will be stored in the hash-table of that author.

Example:

Book-1: How to Program in C++ by XYZ
Book-2: How to Program in Java by XYZ
Book-3: How to Program in Python by XYZ
Book-4: How to Program in C++ by ABC
Book-5: How to Program in MATLAB by ABC



ID	Author
0	XYZ
1	ABC

ID	Book Title
0	How to Program in C++
1	How to Program in Java
2	How to Program in Python

ID	Book Title
0	How to Program in C++
1	How to Program in MATLAB

In the above example, you have to insert the authors in a separate hash table. In front of author XYZ, there will be another hash table that contains the books of that author. In simple words, you have to link another hash table in front of authors' name.

You can perform linear/quadratic/cubic probing in case of collisions. Make sure that your system does not lead to primary clustering.

Note: Chaining is not allowed in this question.

Tasks:

1) Design hash-function(s) to store the authors name and book names at the correct position (Hash-functions with minimum number of collisions will get more marks). Choose the sizes of the hash tables wisely.
2) Count the number of collisions for each table.
3) Handle the collision cases with probing.
4) Your program must be able to perform insertion and deletion.

[illegible]

[illegible]

5) Your program must also be able to perform different types of searching.

   a. For example, program should be able to display all books of a particular author. The program must also be able to search book if the title is given. It must display all the books with same title and with different author's names.

      Searching Example:

         Query: Author Name is ABC

         Results (2 books): How to Program in C++, and How to Program in MATLAB

         Query: Book name is How to Program in C++

         Results (2 authors): This book is available with two authors names: ABC, and XYZ

6) Make a proper menu so that admin can use the system `efficiently.

Note: Books data is given. Use this data. Any other data used by the program may lead to zero marks.