Faraz Rashid | 21I-0659

Ahmad Hassan | 21I-0403

Data Structures Project

A SIMPLE DATABASE SYSTEM

- Class node
- Class AVL
- Class RedBlack
- Struct elements
- Class Bnode
- Class Btree

Class node:

This is a templated class which is used for both AVL and Red Black Trees. It contains the instances of the variables found in the record e.g cause of death, year, ID etc. The node contains a pointer to point towards the right and left sub-childs, as well as a parent pointer(for red black). It contains friend functions to write AVL and red black tree in binary files.

Class AVL:

A templated class which composes a class node pointer root. The root is the first element of the tree. The AVL tree is created using two types of rotations, right and left, which keep the tree in balance so that the complexity of traversing through it is log(n). The AVL class is used in the program to perform insertion, deletion and updation in AVL tree as required by the database system.

Class RedBlack:

A templated class which composes a class node pointer of root and a NULL pointer for its leaf nodes. The root is the first element of the tree and its color is black. The RedBlack tree is created using color coding of red and black, which keeps the tree in balance so that the complexity of traversing through it is log(n). The function, void colorArrangement(node<T>* nodeptr), keeps in order the colors in the Red Black tree.The RedBlack class is used in the program to perform insertion, deletion and updation in the RedBlack tree as required by the database system.

Struct Elements:

In order to store the data most efficiently, we decided to store the various data related to the records in a structure as B trees use a linked list/array to keep track of the various data in a node and instead of making numerous amounts of different data type arrays, we decided to make an array of a struct to be used by the Bnodes.

This structure is a templatised based structure used to store the ID, year, deaths etc. It contains a default constructor, a parameterised constructor, and an = operator which is overloaded for ease of use. Besides this we have decided to overload the << and >> operators to easily display the elements.

Class Bnode:

Another templated class that we decided to use. It is made to be a friend function with the B Tree so it can access its functions. It contains an array of the struct elements which we named as keys, a double * array of children pointers, a bool variable to maintain a record if a node is a leaf or not, as well as a counter variable to check the current amount of keys, and a total amount of keys variable we called minDegree. Besides this we have a default constructor, a parameterised constructor, which is handled by the B tree and any insertion functions, a function to update a node with keys if it is not full, a function to split a

node if it is full. Along with this, we utilize some key utility functions to traverse the nodes, update, and retrieve keys and nodes.

Class BTree:

A templated class which contains a Bnode type pointer called root, as well as an integer variable to store the m order of the tree. When the tree is initialized it asks the user to input the order of the tree and uses a while loop to make sure it is greater than 3. Other than this it contains functions to insert new nodes, to traverse the tree, to seek a certain node, as well as update nodes in the Tree.