LABORATORIES EXERCISE NUMBER 3

INDEX NUMBER: 293071

NAME: TOMASZ MIAZGA

GROUP: 103

1. FILES

The whole project consists of four files:

- Dictionary.h
- Testing.h
- Testing.cpp
- main.cpp

First file contains Dictionary class with its own custom Iterator class inside. In Testing.h Testing class is declared. It's methods are using methods form Dictionary class in order to test them. Testing.cpp has the definitions of those methods written in it. In the last file, main.cpp object of class Testing is initialized and it 'uses' its all of its methods in order to perform tests.

2. DICTIONARY CLASS

Its purpose is to create an AVL tree data structure and provide methods to modify it and an iterator. The trees are built with structure Node objects:

```
        struct Node {

                Node * right = nullptr;
                Node * left = nullptr;
                Node * parent = nullptr;

                Key key;
                Info info;

                short int balance = 0;

                Node(Key key, Info info) : key(key), info(info) {};
                Node(Key key, Info info, int balance) : key(key), info(info),
balance(balance) {};
        };


        Node * top = nullptr;
```

The ‘top’ pointer is already declared in class Node pointer which grants access to the tree. The Node structure comes with two types of constructors that help in dynamic allocation of following Nodes.

Private methods of the class:

```
void recDisplay(Node *, int);
void recDestroy(Node *&);
void recDeleteNode(Node *&, Key, Dictionary &);
bool recInsert(Node *&, Key, Info);
bool recContainsKey(Node *, Key);
int recHeight(Node *);
int recSize(Node *);


bool compareTrees(Node*, Node*);

void getParents(Node *&);

Node * copyTree(Node *);

Node * find(Node *, Key);

bool checkKeyInfoType();


void L_rotation(Node *&);
void R_rotation(Node *&);
void RL_rotation(Node *&);
void LR_rotation(Node *&);
```

Methods that names’ begin with ‘rec’ are recursive versions used in public version of them. Furthermore 4 types of rotation methods are added. The capital letter stands for type of rotation: L for left rotation, LR for left-right rotation etc.

Other methods:

- `bool compareTrees(Node*, Node*)` – compares two subtrees under two given node pointers
- `void getParents(Node *&)` – grants a parent pointer to every node in a tree. Is used after each insertion of a new node
- `Node * copyTree(Node *)` – returns a pointer to a tree copied from a subtree of a passed node
- `Node * find(Node *, Key)` – returns a pointer to node with a passed key. Node passed is used in recursion
- `bool checkKeyInfoType()` – method used in createRandomNodes(int) method. It’s purpose is to check if Key and Info are of type int

Public methods:

```
Dictionary() = default;
Dictionary(int)
Dictionary(const Dictionary<Key, Info> &)
~Dictionary()

Dictionary& operator=(const Dictionary &);


void insert(Key, Info);

void display();

void displayNode(Key);

void destroy();

void deleteNode(Key);

void createRandomNodes(int);

bool containsKey(Key);

bool isEmpty();

int height();

int size();


bool operator==(const Dictionary &);
```

The most extraordinary methods:

- `void createRandomNodes(int)` – method used solely in testing. It creates a given number of nodes in the tree. The 'key' and 'info' values are randomly generated from range 1 to 1000. This method may only be used when the created object of the Dictionary class is of type <int, int>. In other instance the method won't do anything apart from printing a message
- `Dictionary(int)` – a constructor that uses the method mentioned above. The given integer is passed to the method

Additionally, class Iterator is declared inside the Dictionary class. It allows user to use iterators during operating on Dictionary objects (however the methods don't use the iterators alone). Inly one operator (++) is overloaded. There are four methods:

- `Iterator begin()` – returns iterator which points to the bottom-left node of tree
- `Void display()` – is used to display content of iterator
- `Key& getKey()` – returns reference to key
- `Info& getInfo()` – returns reference to info

3. TESTING CLASS

Class is supposed to test methods from Dictionary class. The methods and Testing class itself is declared in the Testing.h file, and the method definitions in Testing.cpp file. The methods:

```
void insertTest();
void displayTest();
void costructorsTest();
void deletingNodesTest();
void containsKeyTest();
void sizeTest();
void operatorsTest();
void iteratorTest();
```