**Tree.h:**

#ifndef TREE\_H

#define TREE\_H

#include "node.h"

#include <string>

using namespace std;

class tree {

public:

int count = 0;

virtual void addNode(long long int iKey, int iRowValue) = 0;

virtual void deleteNode(long long int iKey) = 0;

virtual int findNode(long long int iKey) = 0;

virtual void print() = 0;

};

#endif

**Node.h**

#pragma once

#ifndef NODE\_H

#define NODE\_h

class Node {

public:

long long int iKey;

int iRowNumber;

Node(long long int iKey, int iRowNumber) {

this->iKey = iKey;

this->iRowNumber = iRowNumber;

}

};

#endif

**coloredNode.h**

#ifndef COLOREDNODE\_H

#define COLOREDNODE\_H

using namespace std;

#include "node.h"

class ColoredNode : public Node {

public:

int iColor;

ColoredNode\* parent;

ColoredNode\* left;

ColoredNode\* right;

ColoredNode(long long int iKey, int iRowNumber, int iColor, ColoredNode\* parent) : Node(iKey, iRowNumber) {

this->iColor = iColor;

this->parent = parent;

if (iRowNumber != -1) {

left = new ColoredNode(0, -1, 0, this);

right = new ColoredNode(0, -1, 0, this);

}

else {

left = nullptr;

right = nullptr;

}

}

void ValueSwap(ColoredNode\* second) {

this->iKey = second->iKey;

this->iRowNumber = second->iRowNumber;

}

void deleteNode() {

this->iRowNumber = -1;

this->left = nullptr;

this->right = nullptr;

this->iColor = 0;

}

~ColoredNode() {

delete left;

left = nullptr;

delete right;

right = nullptr;

}

};

#endif // !COLOREDNODE\_H

**binaryNode.h**

#pragma once

#ifndef BINARYNODE\_H

#define BINARYNODE\_H

#include "node.h"

class BinaryNode : public Node {

public:

BinaryNode\* left;

BinaryNode\* right;

BinaryNode(long long int iKey, int iRowNumber) : Node(iKey, iRowNumber) {

left = nullptr;

right = nullptr;

}

void oneWaySwap(BinaryNode\* second) {

this->iKey = second->iKey;

this->iRowNumber = second->iRowNumber;

this->left = second->left;

this->right = second->right;

}

void ValueSwap(BinaryNode\* second){

this->iKey = second->iKey;

this->iRowNumber = second->iRowNumber;

}

void swap(BinaryNode\* second) {

long long int iKeySecond = second->iKey;

int iRowNumber = second->iRowNumber;

BinaryNode\* leftSecond = second->left;

BinaryNode\* rightSecond = second->right;

second->iKey = this->iKey;

second->iRowNumber = this->iRowNumber;

second->left = this->left;

second->right = this->right;

this->iKey = iKeySecond;

this->iRowNumber = iKeySecond;

this->left = leftSecond;

this->right = rightSecond;

}

~BinaryNode() {

delete left;

left = nullptr;

delete right;

right = nullptr;

}

};

#endif // !NODE\_H

**basicNotion.h**

#ifndef BASICNOTION\_H

#define BASICNOTION\_H

using namespace std;

#include <string>;

struct notion {

string FIO;

double GPA;

bool excluded;

notion(string FIO, double GPA, bool excluded) {

this->FIO = FIO;

this->GPA = GPA;

this->excluded = excluded;

}

};

#endif // ! BASICNOTION\_H