```
/*Dylan Jeffers
 *Tahmid Rahman
 *founders of RahmROMJeffers ltd.
 *
 *cheatDetector implementation - compares documents to each
 *other and outputs how many phrases of a given length match
 */

#include <iostream>
#include <fstream>
#include <stdlib.h>
#include "pair.h"
#include "SPLAVL.h"
#include "SplayTree.h"
#include "AVLTree.h"
#include "library/circularArrayList.h"
#include <time.h>
#include <vector>
#include <cstdlib>
using namespace std;

void makeTree(vector<int> keyList, BST<int, int> * treeList);


int main(int argc, char* argv[]){
    if(argc != 5){
    cerr << "Incorrect number of arguments" << endl;
    cerr << "Usage: randomInput treeSize maxC maxR useAVL" << endl;
    return 1;
    }
    unsigned int keyNum = atoi(argv[1]);
    cout << keyNum << endl;
    cout.flush();
    int maxC = atoi(argv[2]);
    int maxR = atoi(argv[3]);
    int useAVL = atoi(argv[4]);
    int randNum;
    BST<int, int>* tree;
    vector<int> keyList;
    if (useAVL == 0) {
        tree = new SPLAVL<int,int>(maxC, maxR);
    }
    else if (useAVL == 1) {
        tree = new SplayTree<int, int>();
    }
    else {
        tree = new AVLTree<int, int>();
    }
    randNum = rand() % (keyNum*10);
    keyList.push_back(randNum);
    while(keyList.size() < keyNum) {
        for (unsigned int i = 0; i < keyList.size(); i++) {
            if (randNum == keyList[i]) {
                break;
            }
            if (randNum != keyList[i] && (i == keyList.size()-1)) {
                keyList.push_back(randNum);
            }
        }
        randNum = rand() % (keyNum*10);
    }

    makeTree(keyList, tree);
    //main engine to test find
    clock_t t1, t2;
    t1 = clock();
    unsigned int range = 2;
    for (unsigned int i = 1; i <= range; i*=2) {
        //note this for loop can be arbitrarily long
        for (unsigned int j=0; j < keyNum; j++) {
            //considering :((rand() % i) + i) % i
            randNum = rand() % i;
            tree->find(keyList[randNum]);
        }
    }
    t2 = clock();
    cout << useAVL << " " << ((float)(t2-t1))/CLOCKS_PER_SEC << endl;
    delete tree;
    return 0;
}

void makeTree(vector<int> keyList, BST<int, int> * tree) {
    for (unsigned int i=0; i<keyList.size(); i++) {
        tree->insert(keyList[i], 0);
    }
}
```