**Homework 6 – CS60 Linnell**

**Arjun Kohli**

**W1579330**

**Problem 1:**

**Main File:**

#include <iostream>

using namespace std;

template <typename T>

// > and - operators must be supported

T abs(T a, T b) {

if ((a-b) > (b-a)) {

return a-b;

} else {

return b-a;

}

}

int main() {

int a = 3;

int b = 2;

cout << abs(a, b) << endl;

}

**Output:**

1

**Problem 2:**

**Header File (p2.h):**

#ifndef P2\_H

#define P2\_H

#include <cmath>

template <typename T>

class tmplt{

private:

T arr[10];

int dimension;

public:

tmplt();

tmplt(int dim);

tmplt(int dim, T a[]);

int dimension\_() { return dimension;}

T index(int i) { return arr[i];}

void operator =(tmplt<T> d) {

dimension = d.dimension\_();

for (int i = 0; i < dimension; i++) {

arr[i] = d.index(i);

}

}

void operator +=(tmplt<T> d) {

if (dimension < d.dimension\_()) {

dimension = d.dimension\_();

}

for (int i = 0; i < dimension; i++) {

arr[i] = arr[i] + d.index(i);

}

}

};

template <typename T>

tmplt<T>::tmplt() {

dimension = 0;

}

template <typename T>

tmplt<T>::tmplt(int dim) {

dimension = dim;

for(int i = 0; i < 10; i++) {

arr[i] = T();

}

}

template <typename T>

tmplt<T>::tmplt(int dim, T a[]) {

dimension = dim;

for(int i = 0; i < dim; i++) {

arr[i] = a[i];

}

}

template <typename T>

tmplt<T> operator +(tmplt<T> &d1, tmplt<T> &d2) {

int dim = d1.dimension\_() + d2.dimension\_();

T a[10];

tmplt<T> d3(dim, a);

for (int i = 0; i < 10; i++) {

d3.index(i) += d1.index(i);

d3.index(i) += d2.index(i);

}

return d3;

}

template <typename T>

bool operator ==(tmplt<T> &d1, tmplt<T> &d2) {

if (d1.dimension\_() == d2.dimension\_()) {

for (int i = 0; i < 10; i++) {

if (d1.index(i) == d2.index(i)) {

} else {

return false;

}

}

} else {

return false;

}

return true;

}

template <typename T>

int operator -(std::string &s1, std::string &s2) {

return s1[0]-s2[0];

}

template <typename T>

float distance(tmplt<T> &d1, tmplt<T> &d2) {

float sum = 0;

for (int i = 0; i < d1.dimension\_(); i++) {

sum += (d1.index(i)-d2.index(i))\*(d1.index(i)-d2.index(i));

}

return sqrt(sum);

}

#endif

**Class File (p2.cpp):**

template <typename T>

tmplt<T>::tmplt() {

dimension = 0;

}

template <typename T>

tmplt<T>::tmplt(int dim) {

dimension = dim;

for(int i = 0; i < 10; i++) {

arr[i] = T();

}

}

template <typename T>

tmplt<T>::tmplt(int dim, T a[]) {

dimension = dim;

for(int i = 0; i < dim; i++) {

arr[i] = a[i];

}

}

template <typename T>

tmplt<T> operator +(tmplt<T> &d1, tmplt<T> &d2) {

int dim = d1.dimension\_() + d2.dimension\_();

T a[10];

tmplt<T> d3(dim, a);

for (int i = 0; i < 10; i++) {

d3.index(i) += d1.index(i);

d3.index(i) += d2.index(i);

}

return d3;

}

template <typename T>

bool operator ==(tmplt<T> &d1, tmplt<T> &d2) {

if (d1.dimension\_() == d2.dimension\_()) {

for (int i = 0; i < 10; i++) {

if (d1.index(i) == d2.index(i)) {

} else {

return false;

}

}

} else {

return false;

}

return true;

}

template <typename T>

int operator -(std::string &s1, std::string &s2) {

return s1[0]-s2[0];

}

template <typename T>

float distance(tmplt<T> &d1, tmplt<T> &d2) {

float sum = 0;

for (int i = 0; i < d1.dimension\_(); i++) {

sum += (d1.index(i)-d2.index(i))\*(d1.index(i)-d2.index(i));

}

return sqrt(sum);

}

**Main File:**

#include<iostream>

#include<string>

#include "p2.h"

using namespace std;

int main() {

string r1[] = {"hello", "there"};

string r2[] = {"I", "am", "Sam"};

tmplt<string> a(2, r1);

tmplt<string> b(3, r2);

a += b;

cout << a.index(0) << endl;

cout << a.index(1) << endl;

cout << a.index(2) << endl;

tmplt<string> c = (a+b);

cout << c.index(0) << endl;

cout << c.index(1) << endl;

cout << c.index(2) << endl;

cout << (a==b) << endl;

int p1[] = {1, 2};

int p2[] = {3, 4};

tmplt<int> d(2, p1);

tmplt<int> e(2, p2);

cout << distance(d, e) << endl;

return 0;

}

**Output:**

helloI

theream

Sam

helloI

theream

Sam

0

2.828443

**Problem 3:**

**Main File:**

#include <iostream>

using namespace std;

template <typename T1, typename T2>

//T1 needs to support [], ==

//T2 needs to support <, ++

T1 frequent(T1 a[], T2 size) {

T1 mode = a[0];

int maxCount = 0;

for (T2 i = 0; i < size; i++) {

int count = 0;

for (T2 j = 0; j < size; j++) {

if(i != j && a[i] == a[j]) {

count++;

}

}

if (count > maxCount) {

maxCount = count;

mode = a[i];

}

}

return mode;

}

int main() {

int arr[] = {6, 2, 3, 2, 6, 2};

cout << frequent(arr, 6) << endl;

return 0;

}

**Output:**

2