**Fraser Burch**

**CS 60 Winter**

**Homework #6**

**Warm-Up**

#include <iostream>

#include <string>

using namespace std;

template <typename T>

T abs (T a, T b);

int main(){

double test1 = 15000.00;

double test2 = 10093.03;

int intTest1 = 15;

int intTest2 = 10;

int resultInt = abs(intTest1, intTest2);

cout << "Result of int abs = " <<resultInt << endl;

cout << "" << endl;

double resultDouble = abs(test1, test2);

cout<< "Result of double abs = " << resultDouble << endl;

}

]template <typename T>

//Type variables must have the subtraction and greater than operations

T abs (T a, T b){

T AminusB = a-b;

T BminusA = b-a;

if(AminusB > BminusA){

return AminusB;

}

else{

return BminusA;

}

}

**Output:**

FHosts-MacBook-Pro:Homework 6 fhost$ ./main

Result of int abs = 5

Result of double abs = 4906.97

FHosts-MacBook-Pro:Homework 6 fhost$

**Problem 2:**

//nDPoint.h

#ifndef NDPOINT\_H

#define NDPOINT\_H

#include <cstdlib>

#include <string>

#include <cmath>

template <typename T>

class nDPoint {

private:

T arrayofT[10];

int n;

public:

nDPoint();

nDPoint(int dimension);

nDPoint(int dimension, T a[]);

T getItemAt(int index){return arrayofT[index];}

int getSize(){return n;}

nDPoint<T> operator =(nDPoint<T> arr);

void operator +=(nDPoint<T> arr);

};

template <typename T>

nDPoint<T>::nDPoint(){

n = 0;

}

template <typename T>

nDPoint<T>::nDPoint(int dimension){

n = dimension;

T y = T();

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

arrayofT[i] = y;

}

}

template <typename T>

nDPoint<T>::nDPoint(int dimension, T a[]){

n = dimension;

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

arrayofT[i] = a[i];

}

}

//T must be able to use the = operator

template <typename T>

nDPoint<T> nDPoint<T>::operator =(nDPoint arr){

n = arr.getSize();

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

arrayofT[i] = arr.getItemAt(i);

}

return \*this;

}

//T must be able to use the += operator

template <typename T>

void nDPoint<T>::operator +=(nDPoint arr){

if(n < arr.getSize()){

n = arr.getSize();

}

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

arrayofT[i] += arr.getItemAt(i);

}

//return \*this;

}

//T must be able to use the + operator

template <typename T>

nDPoint<T> operator +(nDPoint<T> a1, nDPoint<T> a2){

nDPoint<T> newNpoint;

newNpoint += a1;

newNpoint += a2;

return newNpoint;

}

//T must be able to use the = operator

template <typename T>

bool operator ==(nDPoint<T> a1, nDPoint<T> a2){

int counter = 0;

if(a1.getSize() == a2.getSize()){

for(int i = 0; i < a1.getSize(); ++i){

if (a1.getItemAt(i) == a2.getItemAt(i)){

counter++;

}

}

}

if(counter == a1.getSize()){

return true;

}

return false;

}

//overloading - for string

int operator - (const std::string& a1, const std::string& a2);

template <typename T>

double distance(nDPoint<T> a1, nDPoint<T> a2){

double toReturn;

double sumToSqrt = 0.0;

T a1H;

T a2H;

if(a1.getSize() == a2.getSize()){

for(int i = 0; i < a1.getSize(); i++){

a1H = (a1.getItemAt(i));

a2H = (a2.getItemAt(i));

sumToSqrt += double(pow((a1H - a2H),2));

}

toReturn = sqrt(sumToSqrt);

}

return toReturn;

}

#endif

MAIN

//main.cpp

#include "nDPoint.h"

#include <iostream>

using namespace std;

int main(){

int arrInt[2] = {1,2};

int arrInt2[2] = {2,1};

nDPoint<int> one = nDPoint<int>(2, arrInt);

nDPoint<int> equalToOne = nDPoint<int>(2, arrInt);

nDPoint<int> two = nDPoint<int>(2, arrInt2);

nDPoint<int> onePlustwo = one+two;

cout<<"Testing B4 '+' Operator" <<endl;

cout<< "("<<onePlustwo.getItemAt(0)<< ", " <<onePlustwo.getItemAt(1) <<")"<<endl;

cout<<"" <<endl;

cout<<"Testing B4 '==' Operator" <<endl;

if(equalToOne == one){

cout<< "The two are equal" <<endl;

}

cout<<"" <<endl;

cout<<"Testing B5 Distance Function for ints" <<endl;

int arr1[2] = {1,1};

int arr2[2] = {2,2};

nDPoint<int> point1 = nDPoint<int>(2, arr1);

nDPoint<int> point2 = nDPoint<int>(2, arr2);

double distanceBet = distance(point1, point2);

cout <<distanceBet <<endl;

cout<<"" <<endl;

cout<<"Testing B5 Distance Function for strings" <<endl;

string arrStr[1] = {"aa"};

string arrStr2[1] = {"bb"};

nDPoint<string> pointStr1 = nDPoint<string>(1, arrStr);

nDPoint<string> pointStr2 = nDPoint<string>(1, arrStr2);

double distanceBetSTR = distance(pointStr1, pointStr2);

cout <<distanceBetSTR <<endl;

return 0;

}

**Output:**

FHosts-MacBook-Pro:Homework 6 fhost$ g++ \*.cpp -o main

FHosts-MacBook-Pro:Homework 6 fhost$ ./main

Testing B4 '+' Operator

(3, 3)

Testing B4 '==' Operator

The two are equal

Testing B5 Distance Function for ints

1.41421

Testing B5 Distance Function for strings

1

FHosts-MacBook-Pro:Homework 6 fhost$

**Problem 3:**

#include <iostream>

#include <string>

using namespace std;

template <typename T1, typename T2>

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

int main(){

int intArr[10] = {1,8,8,4,5,6,7,8,9,10};

int mostFreqInt = frequent(intArr, 10);

cout << mostFreqInt << endl;

string stringArr[8] = {"WHOA","Hey","Hey","Hey","hi","Hey","Hey","Hey"};

string mostFrequentString = frequent(stringArr, 8);

cout << mostFrequentString << endl;

return 0;

}

template <typename T1, typename T2>

//T1 must

//T2 must have + operator, = operator,

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

T2 hitTemp = 0;

T2 hitMax = 0;

T2 maxIndex = 0;

int itsFirst = 0;

for(T2 i = 0; i < size-1; i++){

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

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

if(itsFirst != 0){

hitTemp++;

}

else{

hitMax++;

}

if(hitTemp > hitMax){

hitMax = hitTemp;

maxIndex = i;

}

}

}

itsFirst = 1;

hitTemp = 0;

}

return(a[maxIndex]);

}

**Output:**

FHosts-MacBook-Pro:Problem Three fhost$ ./main

8

Hey

FHosts-MacBook-Pro:Problem Three fhost$