//

// Map.h

// Coen70Lab7

//

// Created by Yousef Zoumot on 2/16/16.

// Copyright (c) 2016 Yousef Zoumot. All rights reserved.

//

#include <utility>

#include <list>

using namespace std;

template < class K, class V > class Map{

list< pair<K,V> > mList;

public:

void insert (K key, V value);

bool contains\_key(K key);

V value\_of(K key);

void remove\_key(K key);

void printVales();

};

template <class K, class V>

bool Map<K,V>:: contains\_key(K key){

typename list< pair<K, V> >:: iterator it;

for(it=mList.begin(); it!= mList.end(); it++){

if(it->first==key)

return true;

}

return false;

}

template <class K, class V>

void Map<K,V>:: insert(K key, V value){

if(contains\_key(key))

return;

mList.push\_back(pair<K,V> (key, value));

return;

}

template <class K, class V>

V Map<K,V>:: value\_of(K key){

typename list< pair<K, V> >:: iterator it;

for(it=mList.begin(); it!= mList.end(); it++){

if(it->first==key)

return it->second;

}

//cout<<"Does not compute... Please put a valid key in...\n";

return NULL;

}

template < class K, class V>

void Map<K,V>:: remove\_key(K key){

/\* this works too but it is messier

typename list< pair<K, V> >:: iterator it;

for(it=mList.begin(); it!= mList.end(); it++){

if(it->first==key)

mList.remove(pair<K,V>(key, it->second));

}

return;\*/

if(contains\_key(key))

mList.remove( pair<K,V>(key,value\_of(key)));

}

template<class K, class V>

void Map<K,V>:: printVales(){

typename list< pair<K, V> >:: iterator it;

for(it=mList.begin(); it!= mList.end(); it++){

cout<<"Key: "<<it->first;

cout<<" Value: "<<it->second<<"\n";

}

cout<<"\n";

return;

}

#include <iostream>

#include "Map.h"

int main(int argc, const char \* argv[]) {

// insert code here...

Map<int, int> map1;

map1.insert(1,2);

map1.insert(2,3);

map1.insert(3,4);

map1.insert(3,4);

map1.insert(3,3);

map1.insert(4,4);

map1.printVales();

map1.remove\_key(2);

map1.printVales();

map1.value\_of(1);

return 0;

}