#include <iostream>

#include <unordered\_map>

#include <stdexcept>

class PerfectHashing {

public:

void insert(double value) {

size\_t h = hashFunction(value);

if (data.find(h) == data.end()) {

data[h] = value;

} else {

throw std::runtime\_error("Collision detected! Hash value already exists.");

}

}

bool search(double value) {

size\_t h = hashFunction(value);

return data.find(h) != data.end() && data[h] == value;

}

void remove(double value) {

size\_t h = hashFunction(value);

auto it = data.find(h);

if (it != data.end() && it->second == value) {

data.erase(it);

} else {

throw std::runtime\_error("Value not found!");

}

}

private:

std::unordered\_map<size\_t, double> data;

size\_t hashFunction(double value) {

// Використовуємо стандартну хеш-функцію для double

return std::hash<double>()(value);

}

};

int main() {

PerfectHashing ph;

try {

ph.insert(3.14);

ph.insert(2.71);

ph.insert(1.618);

std::cout << "Searching for 3.14: " << (ph.search(3.14) ? "Found" : "Not Found") << std::endl;

std::cout << "Searching for 2.71: " << (ph.search(2.71) ? "Found" : "Not Found") << std::endl;

std::cout << "Searching for 1.618: " << (ph.search(1.618) ? "Found" : "Not Found") << std::endl;

std::cout << "Searching for 0.577: " << (ph.search(0.577) ? "Found" : "Not Found") << std::endl;

ph.remove(2.71);

std::cout << "Searching for 2.71 after removal: " << (ph.search(2.71) ? "Found" : "Not Found") << std::endl;

} catch (const std::exception &e) {

std::cerr << e.what() << std::endl;

}

return 0;

}