#include <iostream>

#include <cstring>

class Database {

char\*\* data;

int dataSize;

public:

Database(char\*\* data, int dataSize) {

this->dataSize = dataSize;

this->data = new char\*[dataSize];

for(int i=0; i<this->dataSize; i++) {

this->data[i] = new char[strlen(data[i]) + 1];

strcpy(this->data[i], data[i]);

}

std::cout << "Inside Database" << std::endl;

}

char\*\* getData() const {

return this->data;

}

int getDataSize() const {

return this->dataSize;

}

void alterData(char\*\* newData) {

this->data = newData;

}

~Database() {

for(int i=0; i<this->dataSize; i++) {

delete data[i];

}

delete[] data;

std::cout << "Exit Database" << std::endl;

}

};

class Sortable {

protected:

Database\* db;

public:

Sortable(char\*\* data, int dataSize) {

db = new Database(data, dataSize);

std::cout << "Inside Sortable" << std::endl;

}

virtual ~Sortable() {

delete db;

std::cout << "Exit Sortable" << std::endl;

}

virtual void sort() = 0;

friend std::ostream& operator<<(std::ostream& out, const Sortable& s) {

char\*\* data = s.db->getData();

for (int i = 0; i < s.db->getDataSize(); i++) {

std::cout << data[i] << " ";

}

std::cout << std::endl;

return out;

}

};

class A : public Sortable {

private:

int parition(char\*\* arr, int low, int high) {

char\* pivot = arr[high];

int i = low - 1;

for(int j=low; j<high; j++) {

if (strlen(arr[j]) < strlen(pivot)) {

i++;

std::swap(arr[i], arr[j]);

}

}

std::swap(arr[i + 1], arr[high]);

return i + 1;

}

void f(char\*\* arr, int low, int high) {

if (low < high) {

int pivotIndex = parition(arr, low, high);

f(arr, low, pivotIndex - 1);

f(arr, pivotIndex + 1, high);

}

}

public:

A(char\*\* data, int dataSize) : Sortable(data, dataSize) {}

virtual void sort() {

char\*\* cache = this->db->getData();

f(cache, 0, this->db->getDataSize() - 1);

this->db->alterData(cache);

}

virtual ~A() {

std::cout << "Exit A" << std::endl;

}

};

class B : public Sortable {

private:

int parition(char\*\* arr, int low, int high) {

char\* pivot = arr[high];

int i = low - 1;

for(int j=low; j<high; j++) {

if (strlen(arr[j]) > strlen(pivot)) {

i++;

std::swap(arr[i], arr[j]);

}

}

std::swap(arr[i + 1], arr[high]);

return i + 1;

}

void f(char\*\* arr, int low, int high) {

if (low < high) {

int pivotIndex = parition(arr, low, high);

f(arr, low, pivotIndex - 1);

f(arr, pivotIndex + 1, high);

}

}

public:

B(char\*\* data, int dataSize) : Sortable(data, dataSize) {

std::cout << "Inside B" << std::endl;

}

virtual void sort() {

char\*\* cache = this->db->getData();

f(cache, 0, this->db->getDataSize() - 1);

this->db->alterData(cache);

}

virtual ~B() {

std::cout << "Exit B" << std::endl;

}

};

int main() {

char\* a[] = {"I", "Love", "C#", ",", "Just", "Kidding", "C", "Forever"};

int dataSize = sizeof(a) / sizeof(a[0]);

Sortable\* sortable\_A = new A(a, dataSize);

Sortable\* sortable\_B = new B(a, dataSize);

std::cout << \*sortable\_A;

std::cout << \*sortable\_B;

sortable\_A->sort();

sortable\_B->sort();

std::cout << \*sortable\_A;

std::cout << \*sortable\_B;

delete sortable\_A;

delete sortable\_B;

return 0;

}