#include<iostream>

using namespace std;

class Meal {

int id;

char\* name;

double rating;

char\* category;

double price;

double tax;

public:

static int s\_id;

#pragma region Constructors

Meal() {

id = ++s\_id;

}

Meal(const char\*n,const double&r,const char\*c,const double&p,const double&t)

{

SetId(++s\_id);

SetName(n);

SetRating(r);

SetCategory(c);

SetPrice(p);

SetTax(t);

}

Meal(const Meal& other) {

SetId(other.id);

SetName(other.name);

SetRating(other.rating);

SetCategory(other.category);

SetPrice(other.price);

SetTax(other.tax);

}

Meal(Meal&& other) {

SetId(other.id);

SetName(other.name);

SetRating(other.rating);

SetCategory(other.category);

SetPrice(other.price);

SetTax(other.tax);

if (other.name) {

delete other.name;

other.name = nullptr;

}

if (other.category) {

delete other.category;

other.category = nullptr;

}

}

#pragma endregion

#pragma region Getters

int GetId()const {

return id;

}

char\* GetName()const {

return name;

}

double GetRating()const {

return rating;

}

char\* GetCategory()const {

return category;

}

double GetPrice()const {

return price;

}

double GetTax()const {

return tax;

}

#pragma endregion

#pragma region Setters

void SetId(const int& id) {

this->id = id;

}

void SetName(const char\* n) {

int l = strlen(n);

this->name = new char[l + 1]{};

strcpy\_s(this->name, l + 1, n);

}

void SetRating(const double& r) {

this->rating = r;

}

void SetCategory(const char\* c) {

int l = strlen(c);

this->category = new char[l + 1]{};

strcpy\_s(this->category, l + 1, c);

}

void SetPrice(const double& p) {

this->price = p;

}

void SetTax(const double& t) {

this->tax = t;

}

#pragma endregion

~Meal()

{

delete this->name;

delete this->category;

}

#pragma region Operator Overloading

friend ostream& operator<<(ostream& out, const Meal& obj);

friend istream& operator>>(istream& in, Meal& obj);

#pragma endregion

};

#pragma region Operator Overloading Body

ostream& operator<<(ostream& out, const Meal& obj) {

out << "MEAL INFO" << endl;

out << "Meal id : " << obj.id << endl;

out << "Meal name : " << obj.name << endl;

out << "Meal rating : " << obj.rating << endl;

out << "Meal category : " << obj.category << endl;

out << "Meal price : " << obj.price << "$"<<endl;

out << "Meal tax : " << obj.tax << "%" << endl;

return out;

}

istream& operator>>(istream& in, Meal& obj) {

char\* name = new char[100]{};

cout << "Enter name : ";

in.getline(name, 100);

double rating = 0;

cout << "Enter rating : ";

in >> rating;

in.ignore();

in.clear();

char\* category = new char[100]{};

cout << "Enter category : ";

in.getline(category, 100);

double price = 0;

cout << "Enter price : ";

in >> price;

double tax = 0;

cout << "Enter tax : ";

in >> tax;

obj.SetName(name);

obj.SetRating(rating);

obj.SetCategory(category);

obj.SetPrice(price);

obj.SetTax(tax);

return in;

}

#pragma endregion

int Meal::s\_id = 0;

class Restaurant {

char\* name;

char\* address;

char\* phonenumber;

double avgcost;

Meal\*\* meals;

int count = 0;

public:

#pragma region Getters

char\* GetName()const {

return name;

}

char\* GetAddress()const {

return address;

}

char\* GetPhone()const {

return phonenumber;

}

double GetAvgCost()const {

return avgcost;

}

Meal\*\* GetMeals()const {

return meals;

}

int GetCount()const {

return count;

}

#pragma endregion

#pragma region Setters

void SetName(const char\* name) {

int l = strlen(name);

this->name = new char[l + 1]{};

strcpy\_s(this->name, l + 1, name);

}

void SetAddress(const char\* address) {

int l = strlen(address);

this->address = new char[l + 1]{};

strcpy\_s(this->address, l + 1, address);

}

void SetPhonenumber(const char\* phonenumber) {

int l = strlen(phonenumber);

this->phonenumber = new char[l + 1]{};

strcpy\_s(this->phonenumber, l + 1, phonenumber);

}

private:

void SetAvgCost(const double& avgcost) {

this->avgcost = avgcost;

}

public:

#pragma endregion

#pragma region Helpers

void CalculateAvgCost() {

double total = 0;

for (size\_t i = 0; i < count; i++)

{

total += meals[i]->GetPrice();

}

SetAvgCost(total / count);

}

void AddMeal(const Meal\* meal) {

auto newmeals = new Meal \* [count + 1]{};

for (size\_t i = 0; i < count; i++)

{

newmeals[i] = new Meal(move(\*(meals[i])));

}

newmeals[count] = new Meal(move(\*meal));

++count;

meals = newmeals;

newmeals = nullptr;

}

friend ostream& operator<<(ostream& out, const Restaurant& r);

#pragma endregion

#pragma region Constructors

Restaurant(const char\*name,const char\*address,const char\*phone)

{

SetName(name);

SetAddress(address);

SetPhonenumber(phone);

}

#pragma endregion

};

ostream& operator<<(ostream& out, const Restaurant& r) {

out << "RESTAURANT INFO" << endl;

out << "Name : " << r.GetName() << endl;

out << "Address : " << r.GetAddress() << endl;

out << "Phone : " << r.GetPhone() << endl;

for (size\_t i = 0; i < r.count; i++)

{

out << \*(r.meals[i]);

}

return out;

}

void main() {

Restaurant r("Ela restoran", "Sheki restoran yani", "3456789");

Meal m;

cin >> m;

system("cls");

r.AddMeal(&m);

cin.ignore();

cin.clear();

Meal m2;

cin >> m2;

system("cls");

r.AddMeal(&m2);

cout << r;

}