#include <iostream>

#include <vector>

#include <string>

#include <algorithm>

#include <unordered\_set>

using namespace std;

// Struct to represent a Cake

struct Cake {

string name;

vector<string> ingredients;

double price;

};

// Vector to store cakes

vector<Cake> cakes = {

{"Chocolate", {"Flour", "Sugar", "Cocoa", "Eggs", "Butter", "Almonds"}, 500.0},

{"Vanilla", {"Flour", "Sugar", "Milk", "Eggs", "Butter"}, 500.0},

{"Red Velvet", {"Flour", "Sugar", "Cocoa", "Eggs", "Cream Cheese"}, 800.0},

{"Strawberry", {"Flour", "Sugar", "Eggs", "Butter", "Vanilla", "Baking powder", "Strawberries"}, 500.0},

{"Coffee", {"Flour", "Sugar", "Eggs", "Butter", "Coffee", "Baking powder"}, 500.0},

{"Cheese", {"Flour", "Sugar", "Eggs", "Butter", "Cheese", "Baking powder"}, 700.0},

{"Carrot", {"Flour", "Sugar", "Eggs", "Butter", "Carrots", "Baking powder"}, 600.0},

{"Lemon", {"Flour", "Sugar", "Eggs", "Butter", "Lemon juice", "Lemon zest", "Baking powder"}, 600.0},

{"Butterscotch", {"Flour", "Sugar", "Eggs", "Butter", "Butterscotch", "Baking powder"}, 800.0},

{"Fruit", {"Flour", "Sugar", "Eggs", "Butter", "Nuts", "Candied fruit", "Dried fruit", "Baking powder"}, 900.0},

{"Pineapple", {"Flour", "Sugar", "Eggs", "Butter", "Crushed Pineapple", "Vanilla", "Baking powder"}, 900.0},

{"Caramel", {"Flour", "Sugar", "Eggs", "Butter", "Buttermilk", "Vanilla", "Baking powder"}, 600.0},

{"Coconut", {"Flour", "Sugar", "Eggs", "Butter", "Grinded coconuts", "Baking powder"}, 800.0},

};

// A set to track the cakes that have been shown

unordered\_set<string> shownCakes;

// Function to view all cakes without repeating

void viewCakes() {

if (cakes.empty()) {

cout << "No cakes available.\n";

} else {

for (size\_t i = 0; i < cakes.size(); ++i) {

// Skip the cakes that have already been shown

if (shownCakes.find(cakes[i].name) != shownCakes.end()) {

continue;

}

// Display the cake

cout << i + 1 << ". " << cakes[i].name << " - $" << cakes[i].price << "\nIngredients: ";

for (const auto &ingredient : cakes[i].ingredients) {

cout << ingredient << ", ";

}

cout << "\b\b \n"; // Erase last comma

// Mark this cake as shown

shownCakes.insert(cakes[i].name);

}

}

}

// Modify customizeCake function to return the customized cake

Cake customizeCake() {

shownCakes.clear(); // Reset the shownCakes set for every new cake order

viewCakes();

cout << "\nSelect a cake to customize (1-" << cakes.size() << "): ";

int choice;

cin >> choice;

if (choice < 1 || choice > cakes.size()) {

cout << "Invalid choice.\n";

return Cake(); // Return an empty cake object if invalid

}

Cake &selectedCake = cakes[choice - 1]; // Get the selected cake

cout << "\nSelected Cake: " << selectedCake.name << "\n";

cout << "Ingredients: ";

for (const auto &ingredient : selectedCake.ingredients) {

cout << ingredient << ", ";

}

cout << "\b\b \n"; // Erase last comma

// Allow user to customize the cake (remove or add ingredients)

string response;

cout << "\nWould you like to remove any ingredient? (yes/no): ";

cin >> response;

if (response == "yes") {

cout << "Enter the ingredient to remove: ";

string ingredientToRemove;

cin.ignore();

getline(cin, ingredientToRemove);

auto it = find(selectedCake.ingredients.begin(), selectedCake.ingredients.end(), ingredientToRemove);

if (it != selectedCake.ingredients.end()) {

selectedCake.ingredients.erase(it);

cout << ingredientToRemove << " removed successfully.\n";

} else {

cout << "Ingredient not found.\n";

}

}

cout << "\nWould you like to add any ingredient? (yes/no): ";

cin >> response;

if (response == "yes") {

cout << "Enter the ingredient to add: ";

string ingredientToAdd;

cin.ignore();

getline(cin, ingredientToAdd);

selectedCake.ingredients.push\_back(ingredientToAdd);

cout << ingredientToAdd << " added successfully.\n";

}

cout << "\nCustomization complete. Updated ingredients: ";

for (const auto &ingredient : selectedCake.ingredients) {

cout << ingredient << ", ";

}

cout << "\b\b \n";

return selectedCake;

}

// Function to take an order

void takeOrder(Cake selectedCake) {

cout << "\nDo you want a message on the cake? (yes/no): ";

string response;

cin >> response;

string message;

if (response == "yes") {

cout << "Enter the message: ";

cin.ignore();

getline(cin, message);

cout << "Message: '" << message << "' will be written on the cake.\n";

}

cout << "Your order has been placed successfully, now it's time for the bill...\n";

cout << "\nOrder Summary:\n";

if (!message.empty()) {

cout << "Message: " << message << "\n";

}

cout << "Cake: " << selectedCake.name << "\n";

cout << "Price: $" << selectedCake.price << "\n";

cout << "Thank you for placing your order at our bakery!\n";

cout << "Good luck and enjoy your cake!\n";

}

// Main function

int main() {

cout << "---------------Welcome to DUO's Cake Bakery Shop--------------\n";

char addMoreCakes;

do {

cout << "Here is our menu of different cake flavors:\n\n";

viewCakes();

cout << "\n-----------------------------\n";

// Get the customized cake from customizeCake

Cake selectedCake = customizeCake(); // This will return the customized cake

cout << "-----------------------------\n";

// Now for ordering the cake

takeOrder(selectedCake);

cout << "\nWould you like to add another cake? (yes/no): ";

cin >> addMoreCakes;

} while (addMoreCakes == 'y' || addMoreCakes == 'Y');

cout << "Thank you for visiting DUO's Cake Bakery! Have a great day!\n";

return 0;

}