#include <iostream>

#include <string>

#include <vector>

struct Contact {

std::string name;

std::string phoneNumber;

};

class Phonebook {

private:

std::vector<Contact> contacts;

public:

void addContact(const std::string& name, const std::string& phoneNumber) {

if (phoneNumber.length() != 10 || !isNumber(phoneNumber)) {

std::cout << "Invalid phone number format. Phone number should contain 10 digits." << std::endl;

return;

}

for (auto& contact : contacts) {

if (contact.phoneNumber == phoneNumber) {

std::cout << "A contact with the same phone number already exists." << std::endl;

std::cout << "Merging contacts." << std::endl;

mergeContacts(contact, name, phoneNumber);

return;

}

}

Contact newContact;

newContact.name = name;

newContact.phoneNumber = phoneNumber;

contacts.push\_back(newContact);

std::cout << "Contact added successfully." << std::endl;

}

void mergeContacts(Contact& existingContact, const std::string& name, const std::string& phoneNumber) {

std::cout << "Merging contact with name: " << existingContact.name << std::endl;

std::cout << "Enter new name for merging: ";

std::string newName;

std::cin >> newName;

existingContact.name = newName;

std::cout << "Contact merged successfully." << std::endl;

}

void modifyContact(const std::string& phoneNumber) {

bool found = false;

for (auto& contact : contacts) {

if (contact.phoneNumber == phoneNumber) {

std::cout << "Enter new name: ";

std::cin >> contact.name;

std::cout << "Contact modified successfully." << std::endl;

found = true;

break;

}

}

if (!found) {

std::cout << "Contact not found." << std::endl;

}

}

void deleteContact(const std::string& phoneNumber) {

auto it = contacts.begin();

bool found = false;

while (it != contacts.end()) {

if ((\*it).phoneNumber == phoneNumber) {

it = contacts.erase(it);

std::cout << "Contact deleted successfully." << std::endl;

found = true;

} else {

++it;

}

}

if (!found) {

std::cout << "Contact not found." << std::endl;

}

}

void searchContact(const std::string& name) {

bool found = false;

for (const auto& contact : contacts) {

if (contact.name == name) {

std::cout << "Name: " << contact.name << std::endl;

std::cout << "Phone Number: " << contact.phoneNumber << std::endl;

found = true;

break;

}

}

if (!found) {

std::cout << "Contact not found." << std::endl;

}

}

void displayAllContacts() {

if (contacts.empty()) {

std::cout << "Phonebook is empty." << std::endl;

} else {

std::cout << "Contacts in the phonebook:" << std::endl;

for (const auto& contact : contacts) {

std::cout << "Name: " << contact.name << std::endl;

std::cout << "Phone Number: " << contact.phoneNumber << std::endl;

std::cout << "-----------------------" << std::endl;

}

}

}

private:

bool isNumber(const std::string& str) {

for (char c : str) {

if (!std::isdigit(c)) {

return false;

}

}

return true;

}

};

int main() {

Phonebook phonebook;

int choice;

std::string name, phoneNumber;

while (true) {

std::cout << "Phonebook Management System" << std::endl;

std::cout << "1. Add Contact" << std::endl;

std::cout << "2. Modify Contact" << std::endl;

std::cout << "3. Delete Contact" << std::endl;

std::cout << "4. Search Contact" << std::endl;

std::cout << "5. Display All Contacts" << std::endl;

std::cout << "6. Exit" << std::endl;

std::cout << "Enter your choice: ";

std::cin >> choice;

switch (choice) {

case 1:

std::cout << "Enter name: ";

std::cin >> name;

std::cout << "Enter phone number: ";

std::cin >> phoneNumber;

phonebook.addContact(name, phoneNumber);

break;

case 2:

std::cout << "Enter phone number of the contact to modify: ";

std::cin >> phoneNumber;

phonebook.modifyContact(phoneNumber);

break;

case 3:

std::cout << "Enter phone number of the contact to delete: ";

std::cin >> phoneNumber;

phonebook.deleteContact(phoneNumber);

break;

case 4:

std::cout << "Enter name to search: ";

std::cin >> name;

phonebook.searchContact(name);

break;

case 5:

phonebook.displayAllContacts();

break;

case 6:

std::cout << "Exiting the program." << std::endl;

return 0;

default:

std::cout << "Invalid choice. Please try again." << std::endl;

break;

}

std::cout << std::endl;

}

return 0;

}