#include <iostream>  
#include <vector>  
#include <string>  
#include <map>  
#include <fstream>

// Struct to hold each survey question and its options  
struct Question {  
    std::string questionText;  // Text of the question  
    std::vector<std::string> options;  // Possible answer choices  
};

// Struct to store the user's response to a question  
struct Response {  
    int questionIndex;  // Index of the question  
    int optionIndex;  // Index of the chosen option  
};

// Enum to represent different political parties  
enum PoliticalParty { DEMOCRAT, REPUBLICAN, LIBERTARIAN, INDEPENDENT };

// Map to store the names of the political parties  
std::map<PoliticalParty, std::string> partyNames = {  
    {DEMOCRAT, "Democrat"},  
    {REPUBLICAN, "Republican"},  
    {LIBERTARIAN, "Libertarian"},  
    {INDEPENDENT, "Independent"}  
};

// Class to handle the survey logic  
class Survey {  
private:  
    std::vector<Question> questions;  // List of survey questions  
    std::vector<Response> responses;  // User responses to the questions  
    std::map<PoliticalParty, int> scores;  // Scores for each political party

public:  
    // Constructor to initialize the survey  
    Survey() {  
        // Initialize scores for each party to zero  
        scores[DEMOCRAT] = 0;  
        scores[REPUBLICAN] = 0;  
        scores[LIBERTARIAN] = 0;  
        scores[INDEPENDENT] = 0;

        // Add survey questions  
        questions.push\_back({"What should the government do to help the poor?",  
                             {"1. Make it easier to apply for assistance",  
                              "2. Allow parents to use education funds for charter schools",  
                              "3. Create welfare to work programs",  
                              "4. Nothing"}});

        questions.push\_back({"What is your stance on healthcare?",  
                             {"1. Support universal healthcare",  
                              "2. Support private healthcare options",  
                              "3. Support a mix of both",  
                              "4. No government involvement in healthcare"}});

        questions.push\_back({"What are your views on gun control?",  
                             {"1. Support strict gun control laws",  
                              "2. Support the right to bear arms",  
                              "3. Support background checks",  
                              "4. No restrictions on gun ownership"}});

        questions.push\_back({"How do you view climate change?",  
                             {"1. Support strong environmental regulations",  
                              "2. Support balanced regulations",  
                              "3. Climate change is not a priority",  
                              "4. Do not believe in climate change"}});

        // More questions can be added as needed...  
    }

    // Method to start the survey and collect responses  
    void startSurvey() {  
        // Loop through each question  
        for (int i = 0; i < questions.size(); ++i) {  
            std::cout << questions[i].questionText << std::endl;  
            // Display the options for the current question  
            for (int j = 0; j < questions[i].options.size(); ++j) {  
                std::cout << questions[i].options[j] << std::endl;  
            }  
            int userResponse;  
            std::cin >> userResponse;  // Get the user's response  
            responses.push\_back({i, userResponse - 1});  // Store the response  
            updateScores(i, userResponse - 1);  // Update the scores based on the response  
        }

        guessParty();  // Guess the user's political party based on the scores  
    }

    // Method to update the scores based on user responses  
    void updateScores(int questionIndex, int optionIndex) {  
        // Update scores based on the question and response  
        // Here we can add weights to the scores based on the intensity  
        if (questionIndex == 0) {  
            if (optionIndex == 0) scores[DEMOCRAT] += 3;  // Strong alignment  
            if (optionIndex == 1) scores[REPUBLICAN] += 3;  // Strong alignment  
            if (optionIndex == 2) scores[LIBERTARIAN] += 2;  // Moderate alignment  
            if (optionIndex == 3) scores[INDEPENDENT] += 1;  // Weak alignment  
        }  
        else if (questionIndex == 1) {  
            if (optionIndex == 0) scores[DEMOCRAT] += 3;  
            if (optionIndex == 1) scores[REPUBLICAN] += 2;  
            if (optionIndex == 2) scores[LIBERTARIAN] += 1;  
            if (optionIndex == 3) scores[INDEPENDENT] += 3;  
        }  
        else if (questionIndex == 2) {  
            if (optionIndex == 0) scores[DEMOCRAT] += 3;  
            if (optionIndex == 1) scores[REPUBLICAN] += 3;  
            if (optionIndex == 2) scores[LIBERTARIAN] += 2;  
            if (optionIndex == 3) scores[INDEPENDENT] += 1;  
        }  
        else if (questionIndex == 3) {  
            if (optionIndex == 0) scores[DEMOCRAT] += 3;  
            if (optionIndex == 1) scores[REPUBLICAN] += 2;  
            if (optionIndex == 2) scores[LIBERTARIAN] += 1;  
            if (optionIndex == 3) scores[INDEPENDENT] += 3;  
        }

        // Additional logic for other questions can be added here...  
    }

    // Method to guess the user's political party based on scores  
    void guessParty() {  
        PoliticalParty guessedParty = DEMOCRAT;  
        int maxScore = 0;  
        // Find the party with the highest score  
        for (const auto& score : scores) {  
            if (score.second > maxScore) {  
                maxScore = score.second;  
                guessedParty = score.first;  
            }  
        }

        // Display the guessed party  
        std::cout << "Based on your responses, we guess your political party is: " << partyNames[guessedParty] << std::endl;

        // Ask the user to confirm their actual party affiliation  
        std::cout << "Which political party do you affiliate with?\n1. Democrat\n2. Republican\n3. Libertarian\n4. Independent\n";  
        int actualParty;  
        std::cin >> actualParty;  
        std::cout << "You selected: " << partyNames[static\_cast<PoliticalParty>(actualParty - 1)] << std::endl;

        // Save the response to a file  
        saveResponse(static\_cast<PoliticalParty>(actualParty - 1));  
    }

    // Method to save the response to a file  
    void saveResponse(PoliticalParty party) {  
        std::string fileName = partyNames[party] + ".txt";  
        std::ofstream file;  
        file.open(fileName, std::ios\_base::app);  // Open the file in append mode

        for (const auto& response : responses) {  
            file << "Question: " << questions[response.questionIndex].questionText << "\n";  
            file << "Answer: " << questions[response.questionIndex].options[response.optionIndex] << "\n\n";  
        }

        file.close();  
    }  
};

// Main function to run the survey  
int main() {  
    Survey survey;  // Create a Survey object  
    survey.startSurvey();  // Start the survey  
    return 0;  
}