```cpp
#include <iostream>
#include <string>
#include <vector>
#include <algorithm>
#include <fstream>
#include <cassert>

using namespace std;

// hold words to ignore
vector<string> CommonWords;

// hold statements to say based on user input
vector<string> Statements;

// statements to use if there isn't any subject
vector<string> BlankStatements;


vector<string> getWordsFromLine(string line)
{
        vector<string> parts;

        int prevPos = 0;
        int spacePos = line.find(" ", 0);

        string word;

        while (spacePos != std::string::npos)
        {
                word = line.substr(prevPos, (spacePos-prevPos));
                // remove(word.begin(), word.end(), " ");

                if (!word.empty()) parts.push_back(word);

                prevPos = spacePos + 1;
                spacePos = line.find(" ", prevPos);
        }

        word = line.substr(prevPos, (spacePos-prevPos));
        // remove(word.begin(), word.end(), " ");

        if (!word.empty()) parts.push_back(word);

        return parts;
}

// initial (annoying) chatbot- just parrot everything the user says
void mock(string input)
{
        cout << '"' + input + '"' << "\n";
}

void talkAbout(string subject)
{
        int statementIndex = rand() % Statements.size();

        string statementBase = Statements[statementIndex];
        int subjectLoc = statementBase.find("@@", 0);

        string newString = statementBase.replace(subjectLoc, 2, subject);
        cout << newString << "\n";
}

void sayRandomStatement()
{
        int statementIndex = rand() % BlankStatements.size();
        cout << BlankStatements[statementIndex] << "\n";
}

void talkToUser(string input)
{
        vector<string> parts = getWordsFromLine(input);

        vector<string>::iterator it;

        // find a subject if one exists
        for (int i=0; i<parts.size(); i++)
        {
                it = find(CommonWords.begin(), CommonWords.end(), parts[i]);

                if (it == CommonWords.end())
                {
                        talkAbout(parts[i]);
                        return;
                }
        }

        // if we didn't find a subject, say *something*
        sayRandomStatement();
}

bool processInput(string input)
{
        if (input == "quit" || input == "q") return false;

        talkToUser(input);

        return true;
}


void InitCommonWords()
{
        ifstream wordFile("words.txt");
        assert(wordFile);

        string word;
        while (getline(wordFile, word))
        {
                CommonWords.push_back(word);
        }

        wordFile.close();

        sort(CommonWords.begin(), CommonWords.end());
```

```
}


void InitStatements()
{
        ifstream statementFile("statements.txt");
        assert(statementFile);

        string statement;
        while (getline(statementFile, statement))
        {
                Statements.push_back(statement);
        }

        statementFile.close();

        ifstream blankStatementFile("randomStatements.txt");
        assert(blankStatementFile);

        while (getline(blankStatementFile, statement))
        {
                BlankStatements.push_back(statement);
        }

        blankStatementFile.close();

}

int main()
{
        InitCommonWords();
        InitStatements();

        cout << "Hello, how are you today?\n";
        string input;

        bool IsActive = true;

        while (IsActive)
        {
                getline(cin, input);
                IsActive = processInput(input);
        }


        return 0;
}
```