```cpp
// Comments are written in English, as per instructions. Additionally, in a real
code most if not of these comments will be needed, as a clear

// code, should be almost self-explanatory. Here they are written for the
purposes of the excersice.


// The validation that check if the ownerName is valid has an issue when the
given name also has spaces, so as for the current code to work, only

// names consisting of letters are accepted.Lost over an hour trying to
understand why input like "Parvo vtoro" is received, just as "Parvo" by

// the program.

#include <iostream>

#include <vector>

#include <cctype>

#include <ctype.h>

// Here we include some parts of the standard library, so our code recognizes
their functions.

using namespace std;

// Here we inform the program to associate all keywords related to "std", to it.
Hence eliminating the need to write "std::"" everytime, however turning those
words

// into key words here as well. For example, "cin" cannot be used as a variable's
name in our code now.


class RepairRecord {

    public:

    int recordId;

    string licensePlate;

    string ownerName;

    string repairDescription;

    vector<string> partsUsed;

    int labourHours;

};
```

```
// Here we define the attributes, which will be a part of our class. Setting an
ID as to have a way to distinguish between the objects, in a way

// different then location memory ID (the id of the cell in the RAM, the object
is saved). Additionaly, a vector is set for partsUsed, as not

// to limit the maximum amount of parts that can be added.


void addRepairRecord(int id, vector<RepairRecord> repairRecordLog){

    bool validPlateFlag = false;

    RepairRecord newRecord;

    // Flag used to exit the while loop, demanding a valid plate number.

    newRecord.recordId = id;


    cout << "Enter the car's license plate." << endl;

    while (validPlateFlag == false){

        // We validate that the entered plate is real - based on simple
assumption that plate numbers have beetween 7 and 8 signs and forcing the

        // user to type in plate numbers, until a valid one is written.

        cin >> newRecord.licensePlate;

        if (size(newRecord.licensePlate) > 6 && size(newRecord.licensePlate) < 9)
{

            validPlateFlag = true;

        } else {

            cout << "Enter a valida plate number - should have 7 or 8 signs in
it." <<endl;

        }

    }


    cout << "Enter the name of the car's owner." << endl;

    string owner;

    while (true) {

        cin >> owner;
        if (size(owner) < 4 || size(owner) > 30){
            // If the entered name is not between 4 and 30 signs, forces the user
to type the name again.
            cout << "Enter a valid name with more than 3 and less than 31
letters." << endl;
            continue;
        }
        bool letterFlag = true;
        bool flag = true;
        for (int i = 0; i < size(owner); i++){
            if (isalpha(owner[i]) || isblank(owner[i])){
                continue;   // useless line, but did not have enough time to
figure out how to state:    "if not"
            } else {
                cout << "The name should consist only of letters and spaces." <<
endl;
                letterFlag = false;
                break;      // breaks the for loop, as the validation has failed
and new name is expected
            }
        }
        if (letterFlag == true) {
            newRecord.ownerName = owner;
            break;          // breaks the while loop, as the validation is
succesful and the code can carry on
        } else if (letterFlag == false){
            continue;
        }

        // The for loop checks is any of the sting signs is different than a
letter or a space. If so a message appears and the loop "continues"
```

```
        // to its next iteration, forcing the user to intput a real name.
Otherwise if the while loop passes through the first if and the for loop

        // a "brake" command is initiated to stop it, as it would mean that the
name is valid.

    }


    cout << "Enter a description of the problem that needs to be fixed." << endl;

    cin >> newRecord.repairDescription;


    cout << "Enter the spare parts that will be required to do the repairs. Once
finised, type 'done' to continue with the next step." << endl;

    string sparePart;

    string sparePartLower;

    while (true){

        cin >> sparePart;

        for (int i = 0; i < size(sparePart); i++){

            sparePartLower += tolower(sparePart[i]);

            }

        // Convert the input to all lower case characters, as to allow
comparisons in the following if statements. Not very optimised way,

        // especially since we need to lower just to seek for the exit word
'done'.

        cout << sparePartLower << endl;

        if (sparePartLower == "done"){

            break;

        } else {

            newRecord.partsUsed.push_back(sparePart);

            // Couldnt find a way to go further than this in the time I had.
Conceptually, i was treating the C++ vector as a python list.

            // But there might be differences i have not grasped. As i tried with
both push_back and vector.insert(end(vector), value). In both cases, the result
is the same and I cant figure out why.
```

```
        }

    }


    cout << "Enter amount labour hours needed to finish the repairs." << endl;

    cin >> newRecord.labourHours;

    cout << "The form has been finished." << endl;

    repairRecordLog.push_back(newRecord);

}


void displayRepairRecords(){

}




int main() {

vector<RepairRecord> repairRecordLog;

// Create a vector to store all records. We use vestor as we need a dynamic
array, since we cannot anticipate its size.

bool stopProgram = false;

// Set a flag to allow to stop the program (while loop)

string userInput;

int counter = 0;

// Will be used to give ids to each record. Starts at 0, as to corespond to the
indecies in the vector repairRecordLog

cout << "Welcome to our repair shop." << endl;


while (stopProgram == false) {

    // We use a while loop, to keep the program running as long as the user
wishes

    cout << "Type 'add' to add a new record, 'display' to view all records,
'search' to find all records of a certain vehicle" <<
```

```
    " or 'exit' to stop the program." << endl;

    cin >> userInput;

    string userInputLowered;

    for (int i = 0; i < size(userInput); i++){

        // Convert the input to all lower case characters, as to allow
comparisons in the following if statements.

        userInputLowered += tolower(userInput[i]);

    }

    if (userInputLowered == "add") {

        // We use several if clauses, as to provide the user with the desired
service.

        cout << "Please answer the next few questions, in order to fill in the
new record." << endl;

        addRepairRecord(counter, repairRecordLog);

        counter++;

    }

    else if (userInputLowered == "display") {


    }

    else if (userInputLowered == "search") {


    }

    else if (userInputLowered == "exit") {

        // Allows the user to quit the program

        cout << "Thanks for visiting our store.";

        break;

    }

    else {

        // Informs the user, the command they have entered is invalid and leads
them to trying again, in the next iteration
```

```
        cout << "The command you have entered is invalid. Try again.";
    }
}

return 0;
}
```