

4IT Erasmus+Project 2024-1-RO01-KA121-VET-000214264
Beneficiary Colegiul National Gheorghe Vranceanu, Bacau, Romania

Sending organization: Colegiul National ”Roman-Vodă” Roman, Romania

MOBILITY PORTFOLIO
Paphos, Cyprus

Mobility period:
PAPHOS, CYPRUS, 13.04.2025-27.04.2025 – practice period 14.04.2025-25.04.2025

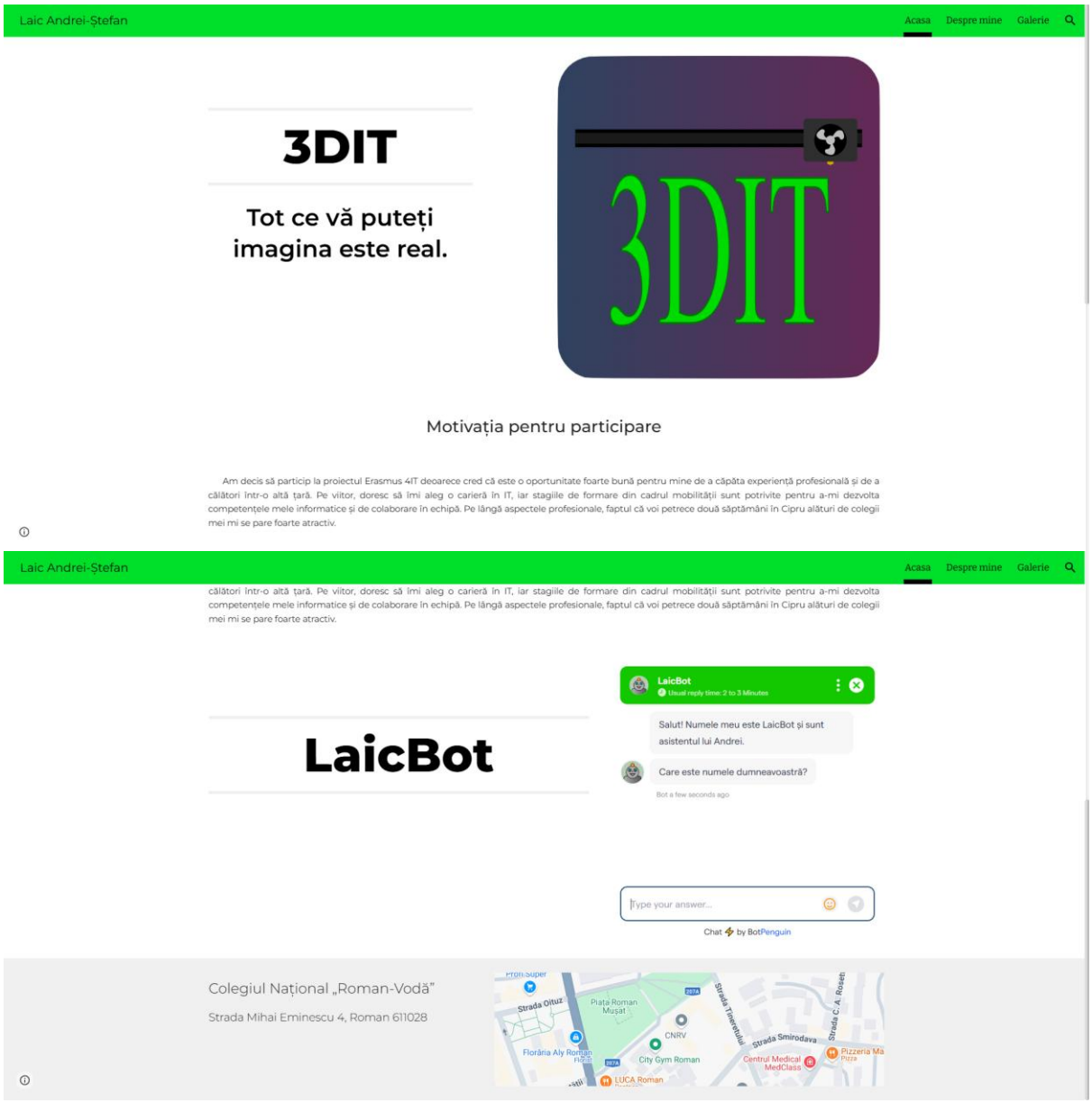
Receiving Lazarou ICT Professionals

Practice tutor: Lazaros Lazarou

Participant: Laic Andrei-Ștefan

DISCLAIMER: The European Commission’s support for the production of this publication does not constitute an endorsement of the contents, which reflect the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

3DIT


Laic Andrei-Ștefan
Acasa
Despre mine
Galerie
3DIT
Tot ce vă puteți imagina este real.
3DIT
Motivația pentru participare
Am decis să particip la proiectul Erasmus 4IT deoarece cred că este o oportunitate foarte bună pentru mine de a căpăta experiență profesională și de a călători într-o altă țară. Pe viitor, doresc să îmi aleg o carieră în IT, iar stagiile de formare din cadrul mobilității sunt potrivite pentru a-mi dezvolta competențele mele informatice și de colaborare în echipă. Pe lângă aspectele profesionale, faptul că voi petrece două săptămâni în Cipru alături de colegii mei mi se pare foarte atractiv.
Laic Andrei-Ștefan
Acasa
Despre mine
Galerie
călători într-o altă țară. Pe viitor, doresc să îmi aleg o carieră în IT, iar stagiile de formare din cadrul mobilității sunt potrivite pentru a-mi dezvolta competențele mele informatice și de colaborare în echipă. Pe lângă aspectele profesionale, faptul că voi petrece două săptămâni în Cipru alături de colegii mei mi se pare foarte atractiv.
LaicBot
LaicBot
Usual reply time: 2 to 3 Minutes
Salut! Numele meu este LaicBot și sunt asistentul lui Andrei.
Care este numele dumneavoastră?
Bot a few seconds ago
Type your answer...
Chat by BotPenguin
Colegiul Național „Roman-Vodă"
Strada Mihai Eminescu 4, Roman 611028
Strada Oituz
Piața Roman Mușat
CNRV
City Gym Roman
Centrul Medical MedClass
LUCA Roman
Strada Smirodava

Despre mine

Informații generale

Mă numesc Laic Andrei-Ștefan, am 17 ani și mă aflu în clasa a XI-a B a Colegiului Național „Roman-Vodă", Roman, profil Matematică-Informatică Intensiv Informatică. În clasele 0-8 am studiat la Școala Gimnazială „Vasile-Alecsandri" unde am avut oportunitatea de a participa la proiectul Erasmus+ ESCEI.

Pasiuni

Pasiunile mele extrașcolare sunt robotica și printarea 3D. Am descoperit domeniul electronicii din gimnaziu prin intermediul platformei Arduino și Raspberry Pi. Acest lucru m-a inspirat să mă înscriu în clasa a IX-a la echipa de robotică a liceului, pe departamentul de programare. Nu a durat foarte mult și au început să mă pasioneze și departamentele de mecanică și proiectare 3D, așa că am achiziționat o imprimantă 3D. Personal, mi se pare foarte interesantă această parte a roboticii, deoarece îți permite să creezi, de la zero, orice obiect îți poți imagina. Cu ajutorul imprimantei, am confecționat atât piese utilizate la concursurile de robotică cât și unele decorative.

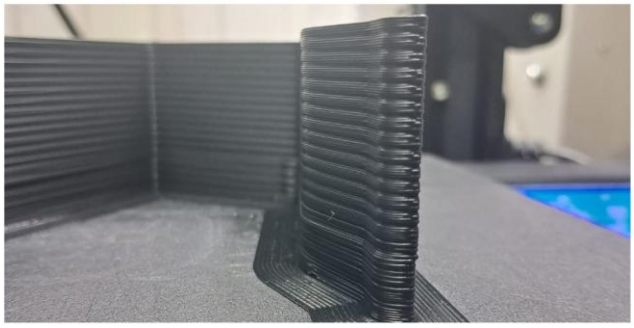
Aspirații școlare

La nivel școlar, sunt pasionat de informatică, fizică și chimie. Mereu încerc să excelez la materiile la care mă pricep, prin participarea la diferite concursuri. De exemplu, în fiecare an școlar m-am calificat la Olimpiada Națională de Fizică iar anul trecut am făcut parte din lotul lărgit de juniori. La Olimpiada Naționala de Chimie am obținut o mențiune MEN în clasa a VIII-a, odată cu o mențiune la fizică, iar în clasa a IX-a am luat Premiul II.

Laic Andrei-Ștefan
Acasa Despre mine Galerie

Galerie Imagini

Imagini imprimantă 3D



Imagini robotică

Laic Andrei-Ștefan
Acasa Despre mine Galerie



C++ Coding Projects

1.

```
#include <iostream>
using namespace std;
/*
This example is showing you how to dynamically grow an array during run
time.
It uses pointers to create the array.
*/

int main(){
    int ARRAYSIZE = 0;
    char* a;
    char* temp;
    int  n = 0, tmpN=0;
    cout<<"How many numbers would you like to enter?:";
    cin>>ARRAYSIZE;
    a = new char[ARRAYSIZE];

    do{
        int x=0;
        cout<<"Enter number "<<n<<" in range 65-90 ->";
        cin >> x;
        while(x<65 || x>90) {
            cout << "Number is not in the range 65-90, please enter
again ->";
            cin >> x;
        }
        a[n] = (char)(x);
        cout<<"Number saved!"<<endl;
        n++;
        if (n >= ARRAYSIZE){                    Y
            cout<<"Out of Space!!!! would you like to enter more? Please
enter how many(0 to Exit): ";
            cin>>tmpN;
            if (tmpN > 0){
                ARRAYSIZE = ARRAYSIZE + tmpN;
                temp = new char[ARRAYSIZE];
                for (int i=0; i<n; i++) {
                    temp[i] = a[i];
                }
                delete [] a;
                a = temp;
            }
        }

    }while (n<ARRAYSIZE);

    cout<<"Now printing the numbers you inputed"<<endl;

    for (int i=0; i<ARRAYSIZE;i++){

        cout<<a[i]<<endl;
```

```
    }
    delete [] a;

    return 0;
}

/*Here you must not delete the temp because we set 'a' to equal temp
        Remember that pointers are variables which actually save memory
addresses.
        in this program, at the part where we grow our array we set
a=temp.
        Therefore a and temp point to the same part of memory. Because
of that you
        need to delete only pointer a, and the memory will be deleted
for temp as well.
*/
```

2.

```
#include <iostream>
using namespace std;

int main()
{
    int *dynamicArray;
    int arrSize = 0;
    cout << "Enter size of the array: ";
    cin >> arrSize;
    dynamicArray = new int[arrSize];
    for(int i = 0; i < arrSize; i++) {
        dynamicArray[i]=i+1;
    }
    cout << '\n';
    for(int i = 0 ; i < arrSize; i++) {
        cout << dynamicArray[i] << " - ";
    }
    cout << '\n';
    delete dynamicArray;
}
```

3.

```
#include <iostream>
#include <string>
using namespace std;

int main () {
    string *nameArray;
    string *telephoneNumberArray;
    int SIZE = 5;
    int n = 0;
    cout << "Please input the number of students: ";
    cin >> SIZE;
```

```
    cin.get();
    nameArray = new string[SIZE];
    telephoneNumberArray = new string[SIZE];

    do {
        cout << "Enter student number " << n+1 << "'s name:\n";
        getline(cin, nameArray[n]);
        cout << "Enter " << nameArray[n] << "'s telephone number:\n";
        getline(cin, telephoneNumberArray[n]);
        n++;
    } while(n<SIZE);

    cout << "\nYou saved: \n-------------------------------------\n";
    for(int i = 0; i < SIZE; i++) {
        cout << "Name and telephone number: " << nameArray[i] << " - "
<< telephoneNumberArray[i] << '\n';
    }
    string nameToSearch;
    cout << "Search a telephone number with a name: ";
    getline(cin, nameToSearch);
    for(int i = 0; i < SIZE; i++) {
        if(nameArray[i].find(nameToSearch) != string::npos) {
            cout << "Found: " << nameArray[i] << " - " <<
telephoneNumberArray[i] << '\n';
        }
    }
}
```

4.

```
#include <iostream>
using namespace std;

int main()
{
    int age;                        // Declaring variables
    char b;
    int *aPtr;                      // Declaring pointers
    char *bPtr;
    age = 5;                        // Setting values
    aPtr = &age;
    cout << age << '\n'; // Result: 5 Outputting to the console
    cout << aPtr << '\n'; // Result: 0x61fe14
    cout << *aPtr << '\n'; // Result: 5

    for(int i=0; i < 10; i++) {
        cout << aPtr << '\n';       // Outputting address
        aPtr++;                     // Incrementing pointer
    }


    return 0;                       // Ending the program
}
```

5.

```
#include <iostream>
#include <fstream>
#include <string>
using namespace std;

int matchScore(string name, string search) {
    int nameLength = name.length();
    int searchLength = search.length();
    int **pd;
    pd = new int*[nameLength+1];
    for(int i = 0; i < nameLength+1; i++) {
        pd[i] = new int[searchLength+1];
    }
    for(int i = 0; i <= nameLength; i++) {
        pd[i][0] = 0;
    }
    for(int j = 1; j <= searchLength; j++) {
        pd[0][j] = 0;
    }

    for(int i = 1; i <= nameLength; i++) {
        for(int j = 1; j <= searchLength; j++) {
            if(name[i-1] == search[j-1]) {
                pd[i][j] = pd[i-1][j-1] + 1;
            } else if(pd[i-1][j] > pd[i][j-1]) {
                pd[i][j] = pd[i-1][j];
            } else {
                pd[i][j] = pd[i][j-1];
            }
        }
    }
    int result = pd[nameLength][searchLength];
    for(int i = 0; i <= nameLength; i++) {
        delete[] pd[i];
    }
    delete[] pd;
    return result;
}

int* searchPhoneWithName(string *nameArray, string
*telephoneNumberArray, int SIZE, string search) {
    int tempSize = SIZE<5?SIZE:5;
    int* bestScores = new int[tempSize];
    int* bestIndexes = new int[tempSize];
    for(int i = 0; i < tempSize; i++) {
        bestScores[i] = -1;
        bestIndexes[i] = -1;
    }
    for(int i = 0; i < SIZE; i++) {
        int score = matchScore(nameArray[i], search);
        int ok = 1;
        for(int j = 0; j < tempSize && ok; j++) {
            if(score > bestScores[j]) {
                for(int k = tempSize - 1; k > j; k--) {
                    bestScores[k] = bestScores[k-1];
                    bestIndexes[k] = bestIndexes[k-1];
```

```
                    ok = 0;
                }
                bestScores[j] = score;
                bestIndexes[j] = i;
            }
        }
    }
    return bestIndexes;
}

int* searchNameWithPhone(string *nameArray, string
*telephoneNumberArray, int SIZE, string search) {
    int tempSize = SIZE<5?SIZE:5;
    int* bestScores = new int[tempSize];
    int* bestIndexes = new int[tempSize];
    for(int i = 0; i < tempSize; i++) {
        bestScores[i] = -1;
        bestIndexes[i] = -1;
    }
    for(int i = 0; i < SIZE; i++) {
        int score = matchScore(telephoneNumberArray[i], search);
        int ok = 1;
        for(int j = 0; j < tempSize && ok; j++) {
            if(score > bestScores[j]) {
                for(int k = tempSize - 1; k > j; k--) {
                    bestScores[k] = bestScores[k-1];
                    bestIndexes[k] = bestIndexes[k-1];
                    ok = 0;
                }
                bestScores[j] = score;
                bestIndexes[j] = i;
            }
        }
    }
    return bestIndexes;
}
ifstream fin("input.txt");
int main () {
    string *nameArray;
    string *telephoneNumberArray;
    int SIZE = 5;
    int n = 0;
    cout << "Please input the number of students: ";
    cin >> SIZE;
    cin.get();
    nameArray = new string[SIZE];
    telephoneNumberArray = new string[SIZE];

    do {
        cout << "Enter student number " << n+1 << "'s name:\n";
        getline(fin, nameArray[n]);
        cout << "Enter " << nameArray[n] << "'s telephone number:\n";
        getline(fin, telephoneNumberArray[n]);
        n++;
    } while(n<SIZE);
```

```
    cout << "You saved:\n--------------------------------------\n";
    for(int i = 0; i < SIZE; i++) {
        cout << "Name and telephone number: " << nameArray[i] << " - "
<< telephoneNumberArray[i] << '\n';
    }
    string searchQuery;
    int option = -1;
    int* bestResults = new int[1];
    do {
        cout << "Enter 0-2 to select operation:\nOption 0 - exit\nOption
1 - search for a telephone number using a name\nOption 2 - search for a
name using a telephone number\nSelect option:";
        cin >> option; cin.get();
        delete[] bestResults;
        switch(option) {
            default:
                cout << "Unknown operation, please try again.\n";
            case 0:
                break;
            case 1:
                cout << "Input the student's name:\n";
                getline(cin, searchQuery);
                bestResults = searchPhoneWithName(nameArray,
telephoneNumberArray, SIZE, searchQuery);
                if(nameArray[bestResults[0]] == searchQuery) {
                    cout << searchQuery << "'s telephone number is: " <<
telephoneNumberArray[bestResults[0]] << '\n';
                } else {
                    cout << "No student named " << searchQuery << " was
found. Here are the " << (SIZE<5?SIZE:5) << " closest results:\n";
                    for(int i = 0; i < (SIZE<5?SIZE:5); i++) {
                        cout << nameArray[bestResults[i]] << " - " <<
telephoneNumberArray[bestResults[i]] << '\n';
                    }
                }
                break;
            case 2:
                cout << "Input the student's telephone number:\n";
                getline(cin, searchQuery);
                bestResults = searchNameWithPhone(nameArray,
telephoneNumberArray, SIZE, searchQuery);
                if(nameArray[bestResults[0]] == searchQuery) {
                    cout << "The student's name associated with
telephone number " << searchQuery << " is: " <<
nameArray[bestResults[0]] << '\n';
                } else {
                    cout << "No student with the telephone number " <<
searchQuery << " was found. Here are the " << (SIZE<5?SIZE:5) << "
closest results:\n";
                    for(int i = 0; i < (SIZE<5?SIZE:5); i++) {
                        cout << nameArray[bestResults[i]] << " - " <<
telephoneNumberArray[bestResults[i]] << '\n';
                    }
                }
                break;
        }
```

```
    } while(option != 0);
    delete[] nameArray;
    delete[] telephoneNumberArray;
}
```

6.

```
#include <iostream>
#include <string>
using namespace std;
class car {
    private:
        string brand;
        string model;
        string plate;
        string owner;

    public:
        car() {
            brand = "Unknown";
            model = "Unknown";
            plate = "Unknown";
            owner = "Unknown";
        }
        void setBrand(string _brand) {
            this->brand = _brand;
        }
        void setModel(string _model) {
            this->model = _model;
        }
        void setPlate(string _plate) {
            this->plate = _plate;
        }
        void setOwner(string _owner) {
            this->owner = _owner;
        }
        string getBrand() {
            return brand;
        }
        string getModel() {
            return model;
        }
        string getPlate() {
            return plate;
        }
        string getOwner() {
            return owner;
        }
};
int matchScore(string name, string search) {
    int nameLength = name.length();
    int searchLength = search.length();
    int **pd;
    pd = new int*[nameLength+1];
    for(int i = 0; i < nameLength+1; i++) {
        pd[i] = new int[searchLength+1];
```

```
    }
    for(int i = 0; i <= nameLength; i++) {
        pd[i][0] = 0;
    }
    for(int j = 1; j <= searchLength; j++) {
        pd[0][j] = 0;
    }

    for(int i = 1; i <= nameLength; i++) {
        for(int j = 1; j <= searchLength; j++) {
            if(name[i-1] == search[j-1]) {
                pd[i][j] = pd[i-1][j-1] + 1;
            } else if(pd[i-1][j] > pd[i][j-1]) {
                pd[i][j] = pd[i-1][j];
            } else {
                pd[i][j] = pd[i][j-1];
            }
        }
    }
    int result = pd[nameLength][searchLength];
    for(int i = 0; i <= nameLength; i++) {
        delete[] pd[i];
    }
    delete[] pd;
    return result;
}
void searchCarByBrand(car *carArray, int NUMBEROFCARS, string search) {
    int* indexes = new int[NUMBEROFCARS];
    int matchCount = 0;
    for(int i = 0; i < NUMBEROFCARS; i++) {
        int score = matchScore(carArray[i].getBrand(), search);
        if(score > 80) {
            indexes[++matchCount] = i;
        }
    }
    indexes[0] = matchCount;
    int *results = indexes;
    cout << results[0] <<" results found:\n";
    for(int i = 1; i <= results[0]; i++) {
        cout << "Car " << results[i] + 1 << ": " <<
carArray[results[i]].getBrand() << " (" <<
carArray[results[i]].getModel() << ")\n";
        cout << "Plate: " << carArray[results[i]].getPlate() << "\n";
        cout << "Owner: " << carArray[results[i]].getOwner() << "\n";
    }
    delete[] indexes;
}
void searchCarByModel(car *carArray, int NUMBEROFCARS, string search) {
    int* indexes = new int[NUMBEROFCARS];
    int matchCount = 0;
    for(int i = 0; i < NUMBEROFCARS; i++) {
        int score = matchScore(carArray[i].getModel(), search);
        if(score > 80) {
            indexes[++matchCount] = i;
        }
    }
```

```
    indexes[0] = matchCount;
    int *results = indexes;
    cout << results[0] <<" results found:\n";
    for(int i = 1; i <= results[0]; i++) {
        cout << "Car " << results[i] + 1 << ": " <<
carArray[results[i]].getBrand() << " (" <<
carArray[results[i]].getModel() << ")\n";
        cout << "Plate: " << carArray[results[i]].getPlate() << "\n";
        cout << "Owner: " << carArray[results[i]].getOwner() << "\n";
    }
    delete[] indexes;
}
void searchCarByPlate(car *carArray, int NUMBEROFCARS, string search,
int MAXSEARCHRESULTS) {
    int tempSize =
NUMBEROFCARS<MAXSEARCHRESULTS?NUMBEROFCARS:MAXSEARCHRESULTS;
    int* bestScores = new int[tempSize];
    int* bestIndexes = new int[tempSize];
    for(int i = 0; i < tempSize; i++) {
        bestScores[i] = -1;
        bestIndexes[i] = -1;
    }
    for(int i = 0; i < NUMBEROFCARS; i++) {
        int score = matchScore(carArray[i].getPlate(), search);
        int ok = 1;
        for(int j = 0; j < tempSize && ok; j++) {
            if(score > bestScores[j]) {
                for(int k = tempSize - 1; k > j; k--) {
                    bestScores[k] = bestScores[k-1];
                    bestIndexes[k] = bestIndexes[k-1];
                    ok = 0;
                }
                bestScores[j] = score;
                bestIndexes[j] = i;
            }
        }
    }
    int *results = bestIndexes;
    if(carArray[results[0]].getPlate() == search) {
        for(int i = 0; i < NUMBEROFCARS; i++) {
            if(carArray[i].getOwner() == search) {
                cout << "Car " << i + 1 << ": " <<
carArray[i].getBrand() << " (" << carArray[i].getModel() << ")\n";
                cout << "Plate: " << carArray[i].getPlate() << "\n";
                cout << "Owner: " << carArray[i].getOwner() << "\n";
            }
        }
    } else {
        cout << "No car with plate number " << search << " was found.
Here are the " <<
(NUMBEROFCARS<MAXSEARCHRESULTS?NUMBEROFCARS:MAXSEARCHRESULTS) << "
closest results:\n";
        for(int i = 0; i <
(NUMBEROFCARS<MAXSEARCHRESULTS?NUMBEROFCARS:MAXSEARCHRESULTS); i++) {
            cout << carArray[results[i]].getBrand() << " (" <<
carArray[results[i]].getModel() << ")\n";
```

```
            cout << "Plate: " << carArray[results[i]].getPlate() <<
"\n";
            cout << "Owner: " << carArray[results[i]].getOwner() <<
"\n";
        }
    }
    delete[] bestScores;
    delete[] bestIndexes;
}
void searchCarByOwner(car *carArray, int NUMBEROFCARS, string search,
int MAXSEARCHRESULTS) {
    int tempSize =
NUMBEROFCARS<MAXSEARCHRESULTS?NUMBEROFCARS:MAXSEARCHRESULTS;
    int* bestScores = new int[tempSize];
    int* bestIndexes = new int[tempSize];
    for(int i = 0; i < tempSize; i++) {
        bestScores[i] = -1;
        bestIndexes[i] = -1;
    }
    for(int i = 0; i < NUMBEROFCARS; i++) {
        int score = matchScore(carArray[i].getOwner(), search);
        int ok = 1;
        for(int j = 0; j < tempSize && ok; j++) {
            if(score > bestScores[j]) {
                for(int k = tempSize - 1; k > j; k--) {
                    bestScores[k] = bestScores[k-1];
                    bestIndexes[k] = bestIndexes[k-1];
                    ok = 0;
                }
                bestScores[j] = score;
                bestIndexes[j] = i;
            }
        }
    }
    int *results = bestIndexes;
    if(carArray[results[0]].getOwner() == search) {
        for(int i = 0; i < NUMBEROFCARS; i++) {
            if(carArray[i].getOwner() == search) {
                cout << "Car " << i + 1 << ": " <<
carArray[i].getBrand() << " (" << carArray[i].getModel() << ")\n";
                cout << "Plate: " << carArray[i].getPlate() << "\n";
                cout << "Owner: " << carArray[i].getOwner() << "\n";
            }
        }
    } else {
        cout << "No car owned by " << search << " was found. Here are
the " << (NUMBEROFCARS<MAXSEARCHRESULTS?NUMBEROFCARS:MAXSEARCHRESULTS)
<< " closest results:\n";
        for(int i = 0; i <
(NUMBEROFCARS<MAXSEARCHRESULTS?NUMBEROFCARS:MAXSEARCHRESULTS); i++) {
            cout << carArray[results[i]].getBrand() << " (" <<
carArray[results[i]].getModel() << ")\n";
            cout << "Plate: " << carArray[results[i]].getPlate() <<
"\n";
            cout << "Owner: " << carArray[results[i]].getOwner() <<
"\n";
```

```
        }
    }
    delete[] bestScores;
    delete[] bestIndexes;
}
car* readCar(car *carArr, int i) {
    car *carArray = carArr;
    string input;
    cout << "------CAR NUMBER " << i + 1 << "------" << '\n';
    cout << "Enter car number " << i + 1 << "'s brand: ";

    getline(cin, input);
    carArray[i].setBrand(input);

    cout << "Enter car number " << i + 1 << "'s model: ";
    getline(cin, input);
    carArray[i].setModel(input);

    cout << "Enter car number " << i + 1 << "'s plate: ";
    getline(cin, input);
    carArray[i].setPlate(input);

    cout << "Enter car number " << i + 1 << "'s owner: ";
    getline(cin, input);
    carArray[i].setOwner(input);

    return carArray;
}
void checkArray(car *&carArray, int &NUMBEROFCARS, int &MAXCARS) {
    if (NUMBEROFCARS >= MAXCARS) {
        MAXCARS += 10;
        car *temp = new car[MAXCARS];
        for (int i = 0; i < NUMBEROFCARS; i++) {
            temp[i] = carArray[i];
        }
        delete[] carArray;
        carArray = temp;
    }
}
void displayCars(car *carArray, int NUMBEROFCARS) {
    cout << "You saved:" << '\n';
    for (int i = 0; i < NUMBEROFCARS; i++) {
        cout << "Car number " << i + 1 << ": " << carArray[i].getBrand()
<< " " << carArray[i].getModel() << '\n';
        cout << "Plate: " << carArray[i].getPlate() << '\n';
        cout << "Owner: " << carArray[i].getOwner() << '\n';
    }
}
int main() {
    int MAXCARS = 50;
    car *carArray;
    int NUMBEROFCARS = 0;
    int MAXSEARCHRESULTS = 5;
    cout << "Enter the number of cars: ";
    cin >> NUMBEROFCARS; cin.get();
    if(MAXCARS < NUMBEROFCARS) {
```

```
        MAXCARS = NUMBEROFCARS;
    }
    carArray = new car[MAXCARS];

    for (int i = 0; i < NUMBEROFCARS; i++) {
        readCar(carArray, i);
    }
    cout << "\n\n";
    displayCars(carArray, NUMBEROFCARS);

    string searchQuery;
    int option = -1;
    do {
        cout << "Enter 0-6 to select operation:\nOption 0 - exit\nOption
1 - enter a car\nOption 2 - display saved cars\nOption 3 - search a car
by brand\nOption 4 - search a car by model\nOption 5 - search a car by
plate number\nOption 6 - search a car by owner\nSelect option:";
        cin >> option; cin.get();
        switch(option) {
            default:
                cout << "Unknown operation, please try again.\n";
            case 0:
                break;
            case 1:
                checkArray(carArray, NUMBEROFCARS, MAXCARS);
                carArray = readCar(carArray, NUMBEROFCARS);
                NUMBEROFCARS++;
                break;
            case 2:
                displayCars(carArray, NUMBEROFCARS);
                break;
            case 3:
                cout << "Brand to search:\n";
                getline(cin, searchQuery);
                searchCarByBrand(carArray, NUMBEROFCARS, searchQuery);
                break;
            case 4:
                cout << "Model to search:\n";
                getline(cin, searchQuery);
                searchCarByModel(carArray, NUMBEROFCARS, searchQuery);
                break;
            case 5:
                cout << "Plate number to search:\n";
                getline(cin, searchQuery);
                searchCarByOwner(carArray, NUMBEROFCARS, searchQuery,
MAXSEARCHRESULTS);
                break;
            case 6:
                cout << "Owner to search:\n";
                getline(cin, searchQuery);
                searchCarByOwner(carArray, NUMBEROFCARS, searchQuery,
MAXSEARCHRESULTS);
                break;
        }
    } while(option != 0);
```

```
    return 0;
}
// EXERCISE
// BREAK THE ABOVE PROGRAM INTO FUNCTIONS
// CREATE A MENU FOR THE USER TO SELECT OPTIONS FOR THE FOLLOWING INFO
// 1. ENTER DATA
// 2. DISPLAY DATA
// 3. SEARCH BY BRAND
// 4. SEARCH BY MODEL
// 5. SEARCH BY OWNER
// 6. SEARCH BY PLATE NUMBER
// 0. EXIT
// CONVERT THE PROGRAM TO DYNAMIC ARRAYS
```

7.

```
#include <iostream>
#include <string>
using namespace std;

class product {
    private:
        string brand;
        string type;
        double price;
        int stock;
        string barcode;

    public:
        void setBrand(string _brand) {
            this->brand = _brand;
        }
        void setType(string _type) {
            this->type = _type;
        }
        void setPrice(double _price) {
            this->price = _price;
        }
        void setStock(int _stock) {
            this->stock = _stock;
        }
        void setBarcode(string _barcode) {
            this->barcode = _barcode;
        }
        string getBrand() {
            return brand;
        }
        string getType() {
            return type;
        }
        double getPrice() {
            return price;
        }
        int getStock() {
            return stock;
        }
```

```
        string getBarcode() {
            return barcode;
        }
};

int matchScore(string name, string search) {
    int nameLength = name.length();
    int searchLength = search.length();
    int **pd;
    pd = new int*[nameLength+1];
    for(int i = 0; i < nameLength+1; i++) {
        pd[i] = new int[searchLength+1];
    }
    for(int i = 0; i <= nameLength; i++) {
        pd[i][0] = 0;
    }
    for(int j = 1; j <= searchLength; j++) {
        pd[0][j] = 0;
    }

    for(int i = 1; i <= nameLength; i++) {
        for(int j = 1; j <= searchLength; j++) {
            if(name[i-1] == search[j-1]) {
                pd[i][j] = pd[i-1][j-1] + 1;
            } else if(pd[i-1][j] > pd[i][j-1]) {
                pd[i][j] = pd[i-1][j];
            } else {
                pd[i][j] = pd[i][j-1];
            }
        }
    }
    int result = pd[nameLength][searchLength];
    for(int i = 0; i <= nameLength; i++) {
        delete[] pd[i];
    }
    delete[] pd;
    return result*1000000/searchLength/10000;
}

int* searchProductByBrand(product *productArray, int SIZE, string
search) {
    int* indexes = new int[SIZE];
    int matchCount = 0;
    for(int i = 0; i < SIZE; i++) {
        int score = matchScore(productArray[i].getBrand(), search);
        if(score > 80) {
            indexes[++matchCount] = i;
        }
    }
    indexes[0] = matchCount;
    return indexes;
}

int* searchProductByType(product *productArray, int SIZE, string search)
{
    int* indexes = new int[SIZE];
```

```
    int matchCount = 0;
    for(int i = 0; i < SIZE; i++) {
        int score = matchScore(productArray[i].getType(), search);
        if(score > 80) {
            indexes[++matchCount] = i;
        }
    }
    indexes[0] = matchCount;
    return indexes;
}

int searchProductByBarcode(product *productArray, int SIZE, string
search) {
    for(int i = 0; i < SIZE; i++) {
        if(productArray[i].getBarcode() == search) {
            return i;
        }
    }
    return -1;
}


int main() {
    const int MAXPRODUCTS = 100;
    product productArray[MAXPRODUCTS];
    int NUMBEROFPRODUCTS = 0;

    cout << "Enter the number of products: ";
    cin >> NUMBEROFPRODUCTS; cin.get();

    for (int i = 0; i < NUMBEROFPRODUCTS; i++) {
        string input;
        double price;
        int stock;

        cout << "Enter product " << i + 1 << "'s brand: ";
        getline(cin, input);
        productArray[i].setBrand(input);

        cout << "Enter product " << i + 1 << "'s type: ";
        getline(cin, input);
        productArray[i].setType(input);

        cout << "Enter product " << i + 1 << "'s price: ";
        cin >> price; cin.get();
        productArray[i].setPrice(price);

        cout << "Enter product " << i + 1 << "'s stock: ";
        cin >> stock; cin.get();
        productArray[i].setStock(stock);

        cout << "Enter product " << i + 1 << "'s barcode: ";
        getline(cin, input);
        productArray[i].setBarcode(input);
    }
```

```
    cout << "\n\nProducts saved:\n";
    for (int i = 0; i < NUMBEROFPRODUCTS; i++) {
        cout << "Product " << i + 1 << ": " <<
productArray[i].getBrand() << " (" << productArray[i].getType() <<
")\n";
        cout << "Price: €" << productArray[i].getPrice() << "\n";
        cout << "Stock Remaining: " << productArray[i].getStock() <<
"\n";
        cout << "Barcode: " << productArray[i].getBarcode() << "\n";
    }

    string searchQuery;
    int option = -1;
    int* results = new int[1];
    int result = 0;
    do {
        cout << "Enter 0-3 to select operation:\nOption 0 - exit\nOption
1 - search for a product by brand\nOption 2 - search for a product by
type\nOption 3 - buy a product using a barcode\nSelect option:";
        cin >> option; cin.get();
        delete[] results;
        switch(option) {
            default:
                cout << "Unknown operation, please try again.\n";
            case 0:
                break;
            case 1:
                cout << "Brand to search:\n";
                getline(cin, searchQuery);
                results = searchProductByBrand(productArray,
NUMBEROFPRODUCTS, searchQuery);
                cout << results[0] <<" results found:\n";
                for(int i = 1; i <= results[0]; i++) {
                    cout << "Product " << i + 1 << ": " <<
productArray[i].getBrand() << " (" << productArray[i].getType() <<
")\n";
                    cout << "Price: $" << productArray[i].getPrice() <<
"\n";
                    cout << "Stock Remaining: " <<
productArray[i].getStock() << "\n";
                    cout << "Barcode: " << productArray[i].getBarcode()
<< "\n";
                }
                break;
            case 2:
                cout << "Type of product to search:\n";
                getline(cin, searchQuery);
                results = searchProductByType(productArray,
NUMBEROFPRODUCTS, searchQuery);
                cout << results[0] <<" results found:\n";
                for(int i = 1; i <= results[0]; i++) {
                    cout << "Product " << i + 1 << ": " <<
productArray[i].getBrand() << " (" << productArray[i].getType() <<
")\n";
                    cout << "Price: €" << productArray[i].getPrice() <<
"\n";
```

```
                    cout << "Stock Remaining: " <<
productArray[i].getStock() << "\n";
                    cout << "Barcode: " << productArray[i].getBarcode()
<< "\n";
                }
                break;
            case 3:
                cout << "Enter the barcode of the product you want to
buy:\n";
                getline(cin, searchQuery);
                result = searchProductByBarcode(productArray,
NUMBEROFPRODUCTS, searchQuery);
                if(result == -1) {
                    cout << "No product with barcode " << searchQuery <<
" was found.\n";
                } else {
                    cout << "The following product was found:\n";
                    cout << "Product " << result + 1 << ": " <<
productArray[result].getBrand() << " (" <<
productArray[result].getType() << ")\n";
                    cout << "Price: €" <<
productArray[result].getPrice() << "\n";
                    cout << "Stock Remaining: " <<
productArray[result].getStock() << "\n";
                    cout << "Barcode: " <<
productArray[result].getBarcode() << "\n";
                    if(productArray[result].getStock() <= 0) {
                        cout << "Sorry, this product is out of
stock.\n";
                    } else {
                        cout << "Confirm purchase? (y/n)\n";
                        char confirm;
                        cin >> confirm; cin.get();
                        if(confirm == 'y' || confirm == 'Y') {

productArray[result].setStock(productArray[result].getStock() - 1);
                            cout << "Purchase successful! Remaining
stock: " << productArray[result].getStock() << "\n";
                        } else {
                            cout << "Purchase cancelled.\n";
                        }
                    }
                }
                break;
        }
    } while(option != 0);

    return 0;
}
```

8.

```
#include <iostream>
#include <string>
using namespace std;
```

```
class Product {
    private:
        string brand;
        string type;
        double price;
        int stock;
        string barcode;

    public:
        void setBrand(string _brand) {
            this->brand = _brand;
        }
        void setType(string _type) {
            this->type = _type;
        }
        void setPrice(double _price) {
            this->price = _price;
        }
        void setStock(int _stock) {
            this->stock = _stock;
        }
        void setBarcode(string _barcode) {
            this->barcode = _barcode;
        }

        string getBrand() {
            return brand;
        }
        string getType() {
            return type;
        }
        double getPrice() {
            return price;
        }
        int getStock() {
            return stock;
        }
        string getBarcode() {
            return barcode;
        }
};

int matchScore(string name, string search) {
    int nameLength = name.length();
    int searchLength = search.length();
    int **pd;
    pd = new int*[nameLength+1];
    for(int i = 0; i < nameLength+1; i++) {
        pd[i] = new int[searchLength+1];
    }
    for(int i = 0; i <= nameLength; i++) {
        pd[i][0] = 0;
    }
    for(int j = 1; j <= searchLength; j++) {
        pd[0][j] = 0;
    }
```

```
    for(int i = 1; i <= nameLength; i++) {
        for(int j = 1; j <= searchLength; j++) {
            if(name[i-1] == search[j-1]) {
                pd[i][j] = pd[i-1][j-1] + 1;
            } else if(pd[i-1][j] > pd[i][j-1]) {
                pd[i][j] = pd[i-1][j];
            } else {
                pd[i][j] = pd[i][j-1];
            }
        }
    }
    int result = pd[nameLength][searchLength];
    for(int i = 0; i <= nameLength; i++) {
        delete[] pd[i];
    }
    delete[] pd;
    return result*1000000/searchLength/10000;
}

void searchProductByBrand(Product *productArray, int SIZE) {
    string search;
    cout << "Enter the brand to search for: ";
    getline(cin, search);

    int* indexes = new int[SIZE];
    int matchCount = 0;
    for(int i = 0; i < SIZE; i++) {
        int score = matchScore(productArray[i].getBrand(), search);
        if(score > 80) {
            indexes[++matchCount] = i;
        }
    }
    indexes[0] = matchCount;
    int* results = indexes;
    cout << results[0] <<" results found:\n";
    for(int i = 1; i <= results[0]; i++) {
        cout << "Product " << results[i] + 1 << ": " <<
productArray[results[i]].getBrand() << " (" <<
productArray[results[i]].getType() << ")\n";
        cout << "Price: €" << productArray[results[i]].getPrice() <<
"\n";
        cout << "Stock Remaining: " <<
productArray[results[i]].getStock() << "\n";
        cout << "Barcode: " << productArray[results[i]].getBarcode() <<
"\n";
    }
    delete [] indexes;
}

void searchProductByType(Product *productArray, int SIZE) {
    string search;
    cout << "Enter the type to search for: ";
    getline(cin, search);

    int* indexes = new int[SIZE];
```

```
    int matchCount = 0;
    for(int i = 0; i < SIZE; i++) {
        int score = matchScore(productArray[i].getType(), search);
        if(score > 80) {
            indexes[++matchCount] = i;
        }
    }
    indexes[0] = matchCount;
    int* results = indexes;
    cout << results[0] <<" results found:\n";
    for(int i = 1; i <= results[0]; i++) {
        cout << "Product " << results[i] + 1 << ": " <<
productArray[results[i]].getBrand() << " (" <<
productArray[results[i]].getType() << ")\n";
        cout << "Price: €" << productArray[results[i]].getPrice() <<
"\n";
        cout << "Stock Remaining: " <<
productArray[results[i]].getStock() << "\n";
        cout << "Barcode: " << productArray[results[i]].getBarcode() <<
"\n";
    }
    delete [] indexes;
}


Product* inputProduct(Product* productArr, int i) {
    Product* productArray = productArr;
    string input;
    double price;
    int stock;

    cout << "Enter product " << i + 1 << "'s brand: ";
    getline(cin, input);
    productArray[i].setBrand(input);

    cout << "Enter product " << i + 1 << "'s type: ";
    getline(cin, input);
    productArray[i].setType(input);

    cout << "Enter product " << i + 1 << "'s price: ";
    cin >> price; cin.get();
    productArray[i].setPrice(price);

    cout << "Enter product " << i + 1 << "'s stock: ";
    cin >> stock; cin.get();
    productArray[i].setStock(stock);
    cout << "Enter product " << i + 1 << "'s barcode: ";
    getline(cin, input);
    productArray[i].setBarcode(input);

    return productArray;
}

Product* checkArray(Product* productArray, int &numberOfProducts, int
&maxProducts) {
    if (numberOfProducts >= maxProducts) {
```

```
        maxProducts += 10;
        Product* temp = new Product[maxProducts];
        for (int i = 0; i < numberOfProducts; i++) {
            temp[i] = productArray[i];
        }
        delete[] productArray;
        productArray = temp;
    }
    return productArray;
}


void displayProducts(Product* productArray, int numberOfProducts) {
    cout << "\nProducts saved:\n";
    for (int i = 0; i < numberOfProducts; i++) {
        cout << "Product " << i + 1 << ": " <<
productArray[i].getBrand() << " (" << productArray[i].getType() <<
")\n";
        cout << "Price: $" << productArray[i].getPrice() << "\n";
        cout << "Stock Remaining: " << productArray[i].getStock() <<
"\n";
        cout << "Barcode: " << productArray[i].getBarcode() << "\n";
    }
}


void buyProduct(Product* productArray, int numberOfProducts) {
    string searchQuery;
    cout << "Enter the barcode of the product to buy: ";
    getline(cin, searchQuery);

    for (int i = 0; i < numberOfProducts; i++) {
        if (productArray[i].getBarcode() == searchQuery) {
            if (productArray[i].getStock() > 0) {
                productArray[i].setStock(productArray[i].getStock() -
1);
                cout << "Purchase successful! Remaining stock: " <<
productArray[i].getStock() << "\n";
            } else {
                cout << "Sorry, this product is out of stock.\n";
            }
            return;
        }
    }
    cout << "No product with the given barcode was found.\n";
}


int main() {
    int MAX_PRODUCTS = 100;
    Product* productArray = new Product[MAX_PRODUCTS];
    int numberOfProducts = 0;
    int option = -1;
    do {
        cout << "\nMenu:\n";
```

```
        cout << "1. Input products\n";
        cout << "2. Display all products\n";
        cout << "3. Search by brand\n";
        cout << "4. Search by type\n";
        cout << "5. Buy a product by barcode\n";
        cout << "0. Exit\n";
        cout << "Enter your choice: ";
        cin >> option; cin.get();

        switch (option) {
            case 1:
                productArray = checkArray(productArray,
numberOfProducts, MAX_PRODUCTS);
                productArray = inputProduct(productArray,
numberOfProducts);
                numberOfProducts++;
                break;
            case 2:
                displayProducts(productArray, numberOfProducts);
                break;
            case 3:
                searchProductByBrand(productArray, numberOfProducts);
                break;
            case 4:
                searchProductByType(productArray, numberOfProducts);
                break;
            case 5:
                buyProduct(productArray, numberOfProducts);
                break;
            case 0:
                cout << "Exiting program.\n";
                break;
            default:
                cout << "Invalid option. Please try again.\n";
        }
    } while (option != 0);
    return 0;
}
```

9.

```
#include <iostream>
#include <string>
using namespace std;

int search(string *array, int size, string toSearch) {
    for(int i = 0; i < size; i++) {
        if(array[i] == toSearch) {
            return i;
        }
    }
    return -1;
}

int main() {
    int arraySize = 0;
```

```
    cout << "How many people do you want to enter: ";
    cin >> arraySize; cin.get();
    string *nameArray = new string[arraySize];
    for(int i = 0; i < arraySize; i++) {
        cout << "Enter name " << i + 1 << ": ";
        getline(cin, nameArray[i]);
    }
    string toSearch;
    cout << "Enter name to search: ";
    getline(cin, toSearch);
    int index = search(nameArray, arraySize, toSearch);
    if(index != -1) {
        cout << "Found " << toSearch << " at position " << index + 1 <<
'\n';
    } else {
        cout << toSearch << " not found" << '\n';
    }
    delete [] array;
    return 0;
}
```

10.

```
#include <iostream>
#include <string>
using namespace std;

// CREATE THE CLASS
class car {
private:
    string brand;
    string model;

public:
    car() {
        brand = "EMPTY";
        model = "EMPTY";
    }
    void setBrand(string B) {
        brand = B;
    }
    void setModel(string M) {
        model = M;
    }
    string getBrand() {
        return brand;
    }
    string getModel() {
        return model;
    }
};
car *readData(car *arr, int &arrSize, int &nextPos);
int main() {

    int ARRAYSIZE = 4;
    int nextAvailPos = 0;
```

```
    car *carsArray;
    carsArray = new car[ARRAYSIZE];

    int choice = 0;
    do
    {
        cout << "Please choose from the menu below:\n";
        cout << "1. Enter Data\n";
        cout << "2. Display Data\n";
        cout << "3. Search By Brand\n";
        cout << "0. Exit\n";
        cin >> choice;
        cin.get();
        switch (choice)
        {
        case 1:
            carsArray = readData(carsArray, ARRAYSIZE, nextAvailPos);
            break;
        case 2:
            break;
        case 3:
            break;
        case 0:
            break;
        }
    } while (choice != 0);
    return 0;
}
// function car* readData(car *, int)
// the function populates the array with information
// from the user
// the functions accepts a pointer to a car object(the array)
// and the size of the array
// the function returns back a pointer to a car object which is the
array
// and is now populated with data.
car *readData(car *arr, int &arrSize, int &nextPos) {
    int numCars = 0; // number of cars user is going to input
    int newSize = 0; // new size of the array which is calculated by
adding arrSize + numCars
    string tstring;
    car *tmp; // temporary car object to copy everything from the car
array and grow the array;
    cout << "How many cars are you entering: ";
    cin >> numCars;
    cin.get();
    for (int i = 0; i < numCars; i++) {
        if (nextPos >= arrSize) { // first check if the array is full
            newSize = arrSize + numCars;
            tmp = new car[arrSize + numCars]; // allocate temporary
space for the new bigger array
            for (int j = 0; j < arrSize; j++) { //copy everything from
the small array to the bigger one
                tmp[j] = arr[j];
            }
            delete[] arr // release the memory from the old array back
```

```
to the OS;
            arr = tmp; // set the pointer of the array to point to the
new allocated space
            arrSize = newSize; // save the newSize into arrSize
        }
        cout << "Enter BRAND for car{" << nextPos << "}";
        getline(cin, tstring);
        arr[nextPos].setBrand(tstring);
        cout << "Enter MODEL for car{" << nextPos << "}";
        getline(cin, tstring);
        arr[nextPos].setModel(tstring);
        nextPos++;
    }
    return arr;
}
```

11.

```
#include <iostream>
#include <string>
using namespace std;

// CREATE THE CLASS
class car
{
private:
    string brand;
    string model;

public:
    car()
    {
        brand = "EMPTY";
        model = "EMPTY";
    }
    void setBrand(string B)
    {
        brand = B;
    }
    void setModel(string M)
    {
        model = M;
    }
    string getBrand()
    {
        return brand;
    }
    string getModel()
    {
        return model;
    }
};
car *readData(car *arr, int &arrSize, int &nextPos);

void displayData(car *arr, int nextPos);
```

```
int searchByBrand(car *arr, int nextPos);

void editByBrand(car *arr, int nextPos);
int main() {
    int ARRAYSIZE = 4;
    int nextAvailPos = 0;
    car *carsArray;
    carsArray = new car[ARRAYSIZE];

    int choice = 0;
    do
    {
        cout << "Please choose from the menu below:\n";
        cout << "1. Enter Data\n";
        cout << "2. Display Data\n";
        cout << "3. Search By Brand\n";
        cout << "0. Exit\n";
        cin >> choice;
        cin.get();
        switch (choice)
        {
        case 1:
            carsArray = readData(carsArray, ARRAYSIZE, nextAvailPos);
            break;
        case 2:
            // because we are growing the array dinamically
            // ARRAYSIZE stores the total size of the array
            // which may contain empty spots
            // we use nextAvailPos because that is the number of filled
spots
            displayData(carsArray, nextAvailPos);
            break;
        case 3:
            searchByBrand(carsArray, nextAvailPos);
            break;
        case 4:
            // edit data inside the array
            // call the search function first and get the index that you
want to change
            // send it to editData
            // the edit function will show what it is
            // and then will give you the option to enter data
            editByBrand(carsArray, nextAvailPos);
            break;
        case 0:
            break;
        }
    } while (choice != 0);
    return 0;
}
void editByBrand(car *arr, int nextPos) {
    int index = searchByBrand(arr, nextPos);
    string tstring;
    cout << "Enter BRAND for car{" << index << "}";
    getline(cin, tstring);
    arr[index].setBrand(tstring);
```

```
    cout << "Enter MODEL for car{" << index << "}";
    getline(cin, tstring);
    arr[index].setModel(tstring);
}

// function car* readData(car *, int)
// the function populates the array with information
// from the user
// the functions accepts a pointer to a car object(the array)
// and the size of the array
// the function returns back a pointer to a car object which is the
array
// and is now populated with data.
car *readData(car *arr, int &arrSize, int &nextPos) {
    int numCars = 0; // number of cars user is going to input
    int newSize = 0; // new size of the array which is calculated by
adding arrSize + numCars
    string tstring;
    car *tmp; // temporary car object to copy everything from the car
array and grow the array;
    cout << "How many cars are you entering: ";
    cin >> numCars;
    cin.get();
    for (int i = 0; i < numCars; i++) {
        if (nextPos >= arrSize) { // first check if the array is full
            newSize = arrSize + numCars;
            tmp = new car[arrSize + numCars]; // allocate temporary
space for the new bigger array
            for (int j = 0; j < arrSize; j++) { //copy everything from
the small array to the bigger one
                tmp[j] = arr[j];
            }
            delete[] arr // release the memory from the old array back
to the OS;
            arr = tmp; // set the pointer of the array to point to the
new allocated space
            arrSize = newSize; // save the newSize into arrSize
        }
        cout << "Enter BRAND for car{" << nextPos << "}";
        getline(cin, tstring);
        arr[nextPos].setBrand(tstring);
        cout << "Enter MODEL for car{" << nextPos << "}";
        getline(cin, tstring);
        arr[nextPos].setModel(tstring);
        nextPos++;
    }
    return arr;
}

//void displayData(car *, int)
//functin displayData(char*, int) will print all data
//that we saved in our object array at any given time
void displayData(car *arr, int nextPos) {
    for(int i = 0; i < nextPos; i++) {
        cout << "----- CAR " << i+1 << " -----\n";
        cout << "Brand: " << arr[i].getBrand() << '\n';
```

```
        cout << "Model: " << arr[i].getModel() << '\n';
    }
}

int searchByBrand(car *arr, int nextPos) {
    string query;
    int index;
    cout << "Enter the brand you want to search for: ";
    getline(cin, query);
    for(int i = 0; i < nextPos; i++) {
        if(arr[i].getBrand() == query) {
            index = i;
            cout << "----- CAR " << i+1 << " -----\n";
            cout << "Brand: " << arr[i].getBrand() << '\n';
            cout << "Model: " << arr[i].getModel() << '\n';
        }
    }
    return index;
}
```

12.

```
#include <Arduino.h>
#define pushButton 2
#define led 12

int brightness = 0;
int fade = 5;
void setup() {5
    // Initialize serial communication at 9600 bits per second
    Serial.begin(9600);
    pinMode(led, OUTPUT);
    pinMode(pushButton, INPUT);
}

void loop() {
    int sensorValue = analogRead(A0);
    Serial.println(sensorValue);


    brightness = brightness + fade;
    if(brightness <= 0 || brightness >= 255) {
        fade = -fade;
    }
    analogWrite(led, brightness);
    int buttonState = digitalRead(pushButton);
    delay(30);
}
```

Finding a job





Ce este scrisoarea de intenție?

O scrisoare de intenție este un **document formal** trimis împreună cu **CV-ul** cu scopul de a explica de ce persoana respectivă este **interesată** de poziția respectivă și de ce ar fi un **candidat potrivit**.



Conținutul scrisorii de intenție

1. Datele de contact
2. Formula de salut
3. Introducerea
4. Corp
5. Încheiere
6. Semnătura

Importanța scrisorii de intenție

- Completează **CV-ul** cu detalii **personale**
- Evidențiază **motivația** și **entuziasmul**
- Face **diferența** între candidați similari

Sfaturi pentru redactare

- **Personalizează** scrisoarea pentru **fiecare job**
- Păstrează un ton **profesionist** și **entuziast**
- Evită clișeele și informațiile **irelevante**
- Concentrează-te pe ce **poți oferi** companiei
- Fii **clar** și **concis**