**CSCI 1411 – Lab 08 – Searching and Sorting Arrays**

**Goals:**

* Searching through arrays
* Sorting arrays with bubble and selection sorts

**Development Environment:** (all students must use Visual Studios)

**Skills:** arrays, searching sorting, functions,

**Reading**: Chap 7, Chap 8

**Deliverables:** 1) This lab with two screen shots 2) lastnameFirstLab08.cpp

**Part I – Skills Practice (10 points)**

* Open a new project in Visual Studios. Call the project Lab08. Don’t forget to uncheck the Pre-compiled headers and Security Development Lifecycle checks. Check the box for an empty project

You will be creating 4 files called lab08a.cpp, Car.h and CarFunctions.h, CarFunctions.cpp

In Car.h. If you start with the working files from a working Lab07, you will have a lot less to type.

Create a file called Car.h under header files

**#ifndef CAR\_H**

**#define CAR\_H**

**#include <string>**

**using namespace std;**

**struct Car**

**{**

**string make;**

**string model;**

**int year;**

**};**

**#endif //CAR\_H**

Remember in your Car.h, you should remove the #pragma once, and add the duplicate guard:

* **Now** create a file called **CarFunctions.h** under header files

**#ifndef CARFUNCTIONS\_H**

**#define CARFUNCTIONS\_H**

**#include "Car.h"**

**Car newCar(istream &in);**

**bool addCar(Car CarToAdd, Car carArray[], int &currentSize, const int MAXSIZE);**

**void listCars(ostream &out, Car carArray[], int &currentSize);**

**void searchCarMake(string make, Car carArray[], int currentSize);**

**void sortCarMake(Car carArray[], int currentSize);**

**void swap(int&a, int &b);**

**void swap(string &a, string &b);**

**#endif**

* Now create a file called **CarFunctions.cpp** under source files

**#include <iostream>**

**#include "Car.h"**

**#include "CarFunctions.h"**

**using namespace std;**

**Car newCar(istream &in)**

**{**

**Car car1;**

**cout << "Make: ";**

**in.ignore();**

**getline(in, car1.make);**

**cout << "\nModel: ";**

**getline(in, car1.model);**

**cout << "\nYear: ";**

**in >> car1.year;**

**cout << endl;**

**return car1;**

**}**

**bool addCar(Car carToAdd, Car carArray[], int &currentSize, const int MAXSIZE)**

**{**

**currentSize++;**

**if (currentSize < MAXSIZE)**

**{**

**carArray[currentSize - 1] = carToAdd;**

**return true; //successful add**

**}**

**else**

**return false;**

**}**

**void listCars(ostream &out, Car carArray[], int &currentSize)**

**{**

**for (int i = 0; i < currentSize; i++)**

**{**

**out << "Car " << i + 1 << endl;**

**out << carArray[i].make << endl;**

**out << carArray[i].model << endl;**

**out << carArray[i].year << endl << endl;**

**}**

**}**

**void searchCarMake(string make, Car carArray[], int currentSize)**

**{**

**bool found = false;**

**for (int i = 0; i < currentSize; i++)**

**{**

**if (carArray[i].make == make)**

**{**

**found = true;**

**cout << "Car at position " << i << endl;**

**cout << carArray[i].make << endl;**

**cout << carArray[i].model << endl;**

**cout << carArray[i].year << endl<<endl;**

**}//if**

**}//for**

**if (!found)**

**cout << "Record not found\n";**

**}**

**void sortCarMake(Car carArray[], int currentSize)**

**{**

**//bubble sort**

**int maxElement;**

**int index;**

**for (maxElement = currentSize - 1; maxElement > 0; maxElement--)**

**{**

**for (index = 0; index < maxElement; index++)**

**{**

**if (carArray[index].make > carArray[index + 1].make)**

**{**

**//swap the entire record (make, model and year)**

**swap(carArray[index].make, carArray[index + 1].make);**

**swap(carArray[index].model, carArray[index + 1].model);**

**swap(carArray[index].year, carArray[index + 1].year);**

**}//if**

**}//for**

**}//for**

**}//swap**

**void swap(int&a, int &b)**

**{**

**int temp = a;**

**a = b;**

**b = temp;**

**}**

**void swap(string &a, string &b)**

**{**

**string temp = a;**

**a = b;**

**b = temp;**

**}**

* Finally, create a file called **lab08a.cpp**

**#include <iostream>**

**#include "Car.h"**

**#include "CarFunctions.h"**

**using namespace std;**

**int main()**

**{**

**const int MAXSIZE = 100;**

**int currentSize = 0;**

**Car car1;**

**Car carArray[MAXSIZE];**

**int menu;**

**string make;**

**bool done = false;**

**while (!done)**

**{**

**cout << "1. Add a new car to the array\n";**

**cout << "2. List out cars\n";**

**cout << "3. Search for a car by Make\n";**

**cout << "4. Sort cars by Make\n";**

**cout << "5. exit\n";**

**cin >> menu;**

**switch (menu)**

**{**

**case 1: car1 = newCar(cin);**

**addCar(car1, carArray, currentSize, MAXSIZE);**

**break;**

**case 2: listCars(cout, carArray, currentSize);**

**break;**

**case 3: cout << "What make do you want to search for? ";**

**cin >> make;**

**searchCarMake(make, carArray, currentSize);**

**break;**

**case 4:**

**sortCarMake(carArray, currentSize);**

**break;**

**case 5: exit(0);**

**break;**

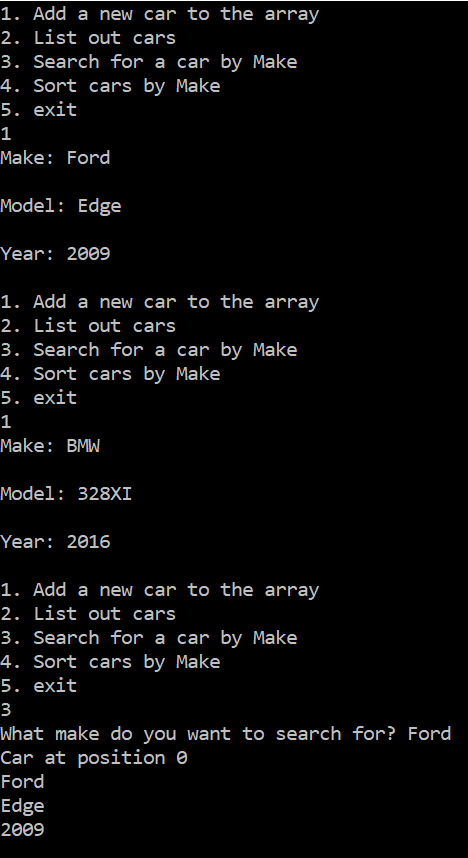
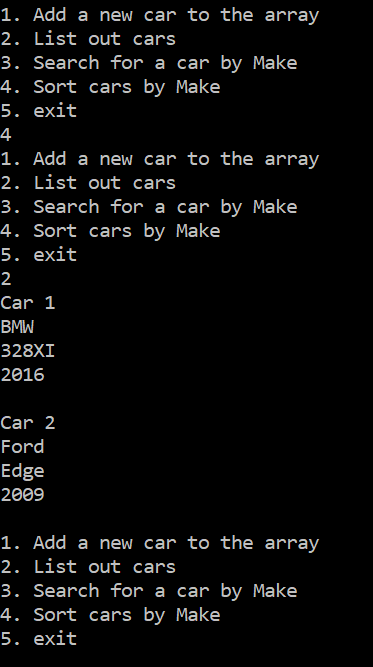
**default: cout << "Number between 1 and 5\n";**

**}//switch**

**}//while**

**}//main**

* **Run** it adding two cars. Take a **screen shot** of the successful output and place it below. Replace this output with your output (and 2 different cars) For a Windows 10 screen shot: Alt key + PrtSc key. Then Ctrl + V to paste. For Mac: Shift + Command + 4. You will not credit unless you have a successful screen shot with Your name in the output.

**Part II - outline/pseudo-code/algorithms (5 pts)**

* Work with your lab partner to write an outline in comments and psuedocode to complete the following program. Use plain English for your outline.
* You should have one file broken into sections with comments. Each .h and .cpp should be in a separate section. Every program you write should have the following block at the top in comments. Make sure to fill in the Name, Class, Description and Lab Partner at the top of the file. Ensure your status is accurate.

/\* Name:  
Class: CSCI 1411-00X  
Description: [fill in description]

Lab Partner:  
Status: successfully compiled and run on csegrid [if it doesn’t run or meet all of the requirements, list the actual status!/\*

**Part III -Movie Listing, Sorting and Searching. (10 pts)**

* Expand on Lab07a, by providing a Search and Sort Function. You should be able to Search or Sort by Title.
* Write a program that reads in data from a file and places the information in a struct named MovieData. The input file has between zero and 100 “records” each with the all of following items (in order, separated as noted below)

Title

ReleaseYear

RunningTime

Rating

Title might have spaces in them. RunningTime is in minutes. Rating is something similar to RottenTomatoes.com, so listed in a percent.

You will have a menu that continues until your press exit, and returns an error and goes back to the menu if not one of the choices. You can assume the user will only enter integers for the menu:

1. Read in Movies
2. Add Movies
3. List Movies
4. Search Movies by Title
5. Sort Movies by Title
6. Exit

For ease of grading, we will have you place all of the code in a single file called yourlastnamefirstLab07.cpp. Your .cpp file should have the following format:

/\* Name:  
Class: CSCI 1411-00X  
Description: [fill in description]

Lab Partner:  
Status: successfully compiled and run on csegrid [if it doesn’t run or meet all of the requirements, list the actual status!/\*

//Function Prototypes

//Function Implementations

int main()

Since they are all together in one file you not have to #include “.h” as we did in the previous lab

Like in the example declare an array in main(). Declare a MAXSIZE in main, then pass the parameters. Remember that you do not pass an array by reference, but if you are going to change the size variable when you Add, that will need to be passed by reference (& before the variable name)

When you add movies, you should add them to the next point in the array and increment the size, but should verify that you are not trying to add past the MAXSIZE of the array.

* Run the program and take a **screenshot** of the output. Place it below
* Turn in yourlastnamefirstLab08.cpp to canvas along with this lab with .