Polytechnic University of Puerto Rico
Electrical and Computer Engineering & Computer Science Department
CECS 2223 – Computer Programming II Lab

Lab 4

Name: Coral S. Schmidt   ID#: 148830

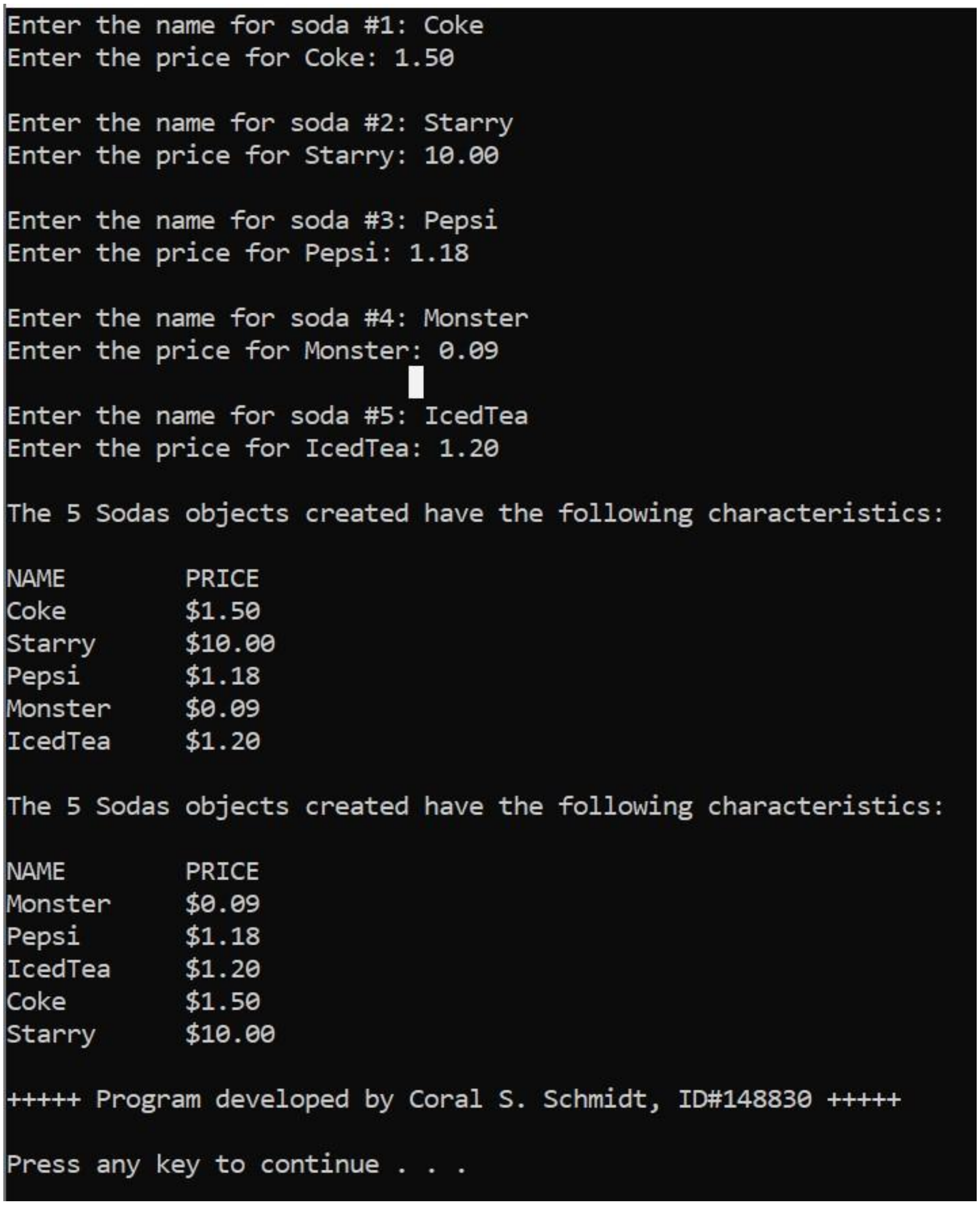
1. Copy the source code developed for Lab 4 and paste it as **text** below. (*15 points*)

```
/*
* CECS 2223, Computer Programming II Laboratory
* Fall 2023, Sec. 09
* Date: September 13, 2023
* Topic: Lab 4 - Arrays of objects and operator overload
* File name: Sodas.h
* This file declares a class named Sodas
* Complete the declaration as required.
* Name: Coral S. Schmidt, ID#148830
*/
#pragma once

// preprocessor directives
#include <string>
#include <iostream>
using namespace std;

class Sodas {
private:
	// declare an integer class variable named sodasCount
	static int sodasCount;

	string name;
	double price;
public:
	// declare the default and copy constructors, and the destructor
	Sodas();
	Sodas(const Sodas& other);
	~Sodas();

	void setName(string);
	void setPrice(double);
	string getName() const;
	double getPrice() const;
	int getSodasCount() const;
	void printSoda() const;

	// declare the overload for the greater than, >, operator
	friend bool operator>(const Sodas& soda1, const Sodas& soda2);
};

/*
* CECS 2223, Computer Programming II Laboratory
* Fall 2023, Sec. 09
* Date: September 13, 2023
* Topic: Lab 4 - Arrays of objects and operator overload
* File name: Sodas.cpp
* Name: Coral S. Schmidt, ID#148830
* This file defines the Sodas class
*/
// do we need any preprocessor directives?
#include "Sodas.h"

// initialize the class variable to 0
int Sodas::sodasCount = 0;

// The default constructor assigns the empty string to the name,
// 0 to the price, and increments the value of sodasCount
Sodas::Sodas()
{
	name = "";
	price = 0.0;
	sodasCount++;
}

// The copy constructor calls the accessor methods from the parameter
// to assign field values to the new object
Sodas::Sodas(const Sodas& other)
{
	name = other.name;
	price = other.price;
}

// The destructor is defined with an empty block of code
Sodas::~Sodas()
{
}

// The set methods DO NOT prompt the user for values or include any
// cout or printf statements. You must validate the parameter's value
// for the price using the ternary operator, assign 0 if invalid.
void Sodas::setName(string sodaName)
{
	name = sodaName;
}

void Sodas::setPrice(double sodaPrice)
{
	price = (sodaPrice >= 0.0) ? sodaPrice : 0.0;
}

// define the get methods, they should all be constant
string Sodas::getName() const
{
	return name;
}

double Sodas::getPrice() const
{
	return price;
```

```
}

int Sodas::getSodasCount() const
{
	return sodasCount;
}



// The printSoda method uses printf to output values formatted
// to be displayed in a table at 12 spaces per column aligned to
// the left of the column. The values to be displayed are: name
// and price. All values must be printed in the same line.
// For example, a Sodas object named Coca-Cola would display:
// Coca-Cola   $1.25
// Make sure to include the dollar sign in the output.
void Sodas::printSoda() const
{
	printf("%-12s $%.2f\n", name.c_str(), price);
}

// The overload to the > operator receives a constant reference to a
// Sodas object as parameter and returns a boolean value. It compares
// the values of the price fields and returns true if the price of the
// object is greater than the price of the parameter, false otherwise.
// This method cannot change the values of the object.

bool operator>(const Sodas& soda1, const Sodas& soda2)
{
	return soda1.price > soda2.price;
}

/*
* CECS 2223, Computer Programming II Lab
* Fall 2023, Sec. 09
* Date: September 13, 2023
* Topic: Lab 4 - Arrays of objects and operator overload
* File name: lab04.cpp
* Name: Coral S. Schmidt, ID#148830
* This file implements the Sodas class using by creating an array
* Complete the C++ code as required
* NOTE: To properly test your code, the array must include at least
* 5 Sodas objects.
*/
// do we need any preprocessor directives?
#include "Sodas.h"

void setSodas(Sodas[], const int);
void arrange(Sodas[], const int);
void displaySodas(const Sodas[], const int);

int main() {
	// declare an integer constant named HOWMANY and initialize it to 5
	const int HOWMANY = 5;
	// declare a Sodas array named mySodas using the constant as the size
	Sodas mySodas[HOWMANY];

	// call the setSodas method
	setSodas(mySodas, HOWMANY);

	// call the displaySodas method
	displaySodas(mySodas, HOWMANY);

	// call the arrange method
	arrange(mySodas, HOWMANY);

	// call the displaySodas method
	displaySodas(mySodas, HOWMANY);

	// write a statement which prints the phrase
	// "+++++ Program developed by [YOUR NAME], ID#[YOUR ID NUMBER] +++++"
	// where the square brackets and the text within is substituted with
	// your personal information. Make sure to include a blank line
	// after the phrase.

	printf("+++++ Program developed by Coral S. Schmidt, ID#148830 +++++\n\n");

	system("pause"); // for Visual Studio use only
	return 0; // this statement MUST be included at the end of main
}

// The setSodas method implements a for iteration control structure to prompt the
// user for the name and price of the sodas in the array, and assigns such values
// to each object. Declare the local string variable named name and initialize it
// to the empty string. Declare a local double variable named price and initialize
it to 0.
// Prompt the user for the name using the phrase "Enter the name for soda #[number]:
",
// where [number] is substituted for the array element. For example, for the first
Sodas
// object in the array, the console would display "Enter the name for soda #1: "
// Similarly, prompt the user for the price for the Sodas object using the phrase
// "Enter the price for [soda name]: " The square brackets must not be printed.
// Make sure to end each cycle with an empty line.
void setSodas(Sodas sodasArray[], const int size)
{
	for (int i = 0; i < size; i++) {
		string name;
		double price;

		printf("Enter the name for soda #%i: ", i + 1);
		cin >> name;

		printf("Enter the price for %s: ", name.c_str());
		cin >> price;

		sodasArray[i].setName(name);
```

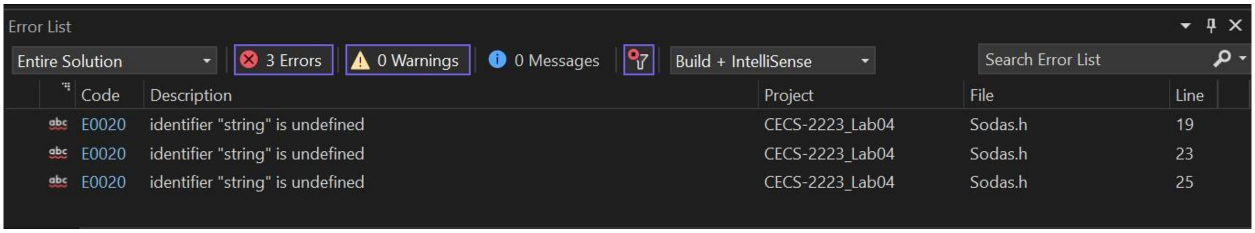
```
        sodasArray[i].setPrice(price);

        printf("\n");
    }
}




// The arrange method receives a Sodas array and its size as parameters and has no
return value.
// It implements a Bubble Sort algorithm to order the values in the array from
lowest to highest.
// This method cannot change the values of any Sodas object.
void arrange(Sodas sodasArray[], const int size)
{
    for (int i = 0; i < size - 1; i++) {
        for (int j = 0; j < size - i - 1; j++) {
            if (sodasArray[j] > sodasArray[j + 1]) {
                Sodas temp = sodasArray[j];
                sodasArray[j] = sodasArray[j + 1];
                sodasArray[j + 1] = temp;
            }
        }
    }
}



// The displaySodas method receives a Sodas array and its size as parameters and has
no return value.
// It prints the values of the Sodas object in a table-like manner. It prints a
phrase, then the
// table headers, and finally the data.
void displaySodas(const Sodas sodasArray[], const int size)
{
    // Substitute [count] for the value of the number of Sodas objects created
    printf("The %d Sodas objects created have the following
characteristics:\n\n", size);

    // table headers, each column is 12 spaces wide and aligned to the left
    printf("%-12s%-12s\n", "NAME", "PRICE");

    // Develop a for control structure to print the state of each Sodas object
    for (int i = 0; i < size; i++) {
        printf("%-12s$%.2f\n", sodasArray[i].getName().c_str(),
sodasArray[i].getPrice());
    }

    // Make sure to add an empty line at the end of the code block
    printf("\n");
}
```

2. Paste the screenshots of the program's execution below. (*5 points*)



```
Enter the name for soda #1: Coke
Enter the price for Coke: 1.50

Enter the name for soda #2: Starry
Enter the price for Starry: 10.00

Enter the name for soda #3: Pepsi
Enter the price for Pepsi: 1.18

Enter the name for soda #4: Monster
Enter the price for Monster: 0.09

Enter the name for soda #5: IcedTea
Enter the price for IcedTea: 1.20

The 5 Sodas objects created have the following characteristics:

NAME        PRICE
Coke        $1.50
Starry      $10.00
Pepsi       $1.18
Monster     $0.09
IcedTea     $1.20

The 5 Sodas objects created have the following characteristics:

NAME        PRICE
Monster     $0.09
Pepsi       $1.18
IcedTea     $1.20
Coke        $1.50
Starry      $10.00

+++++ Program developed by Coral S. Schmidt, ID#148830 +++++

Press any key to continue . . .
```

3. Comment on any warnings or errors revealed by Visual Studio. If any error messages were present, list the error and describe how you corrected it. (*5 points*)



Code	Description	Project	File	Line
E0020	identifier "string" is undefined	CECS-2223_Lab04	Sodas.h	19
E0020	identifier "string" is undefined	CECS-2223_Lab04	Sodas.h	23
E0020	identifier "string" is undefined	CECS-2223_Lab04	Sodas.h	25

I fixed it by implementing

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

in Sodas.h