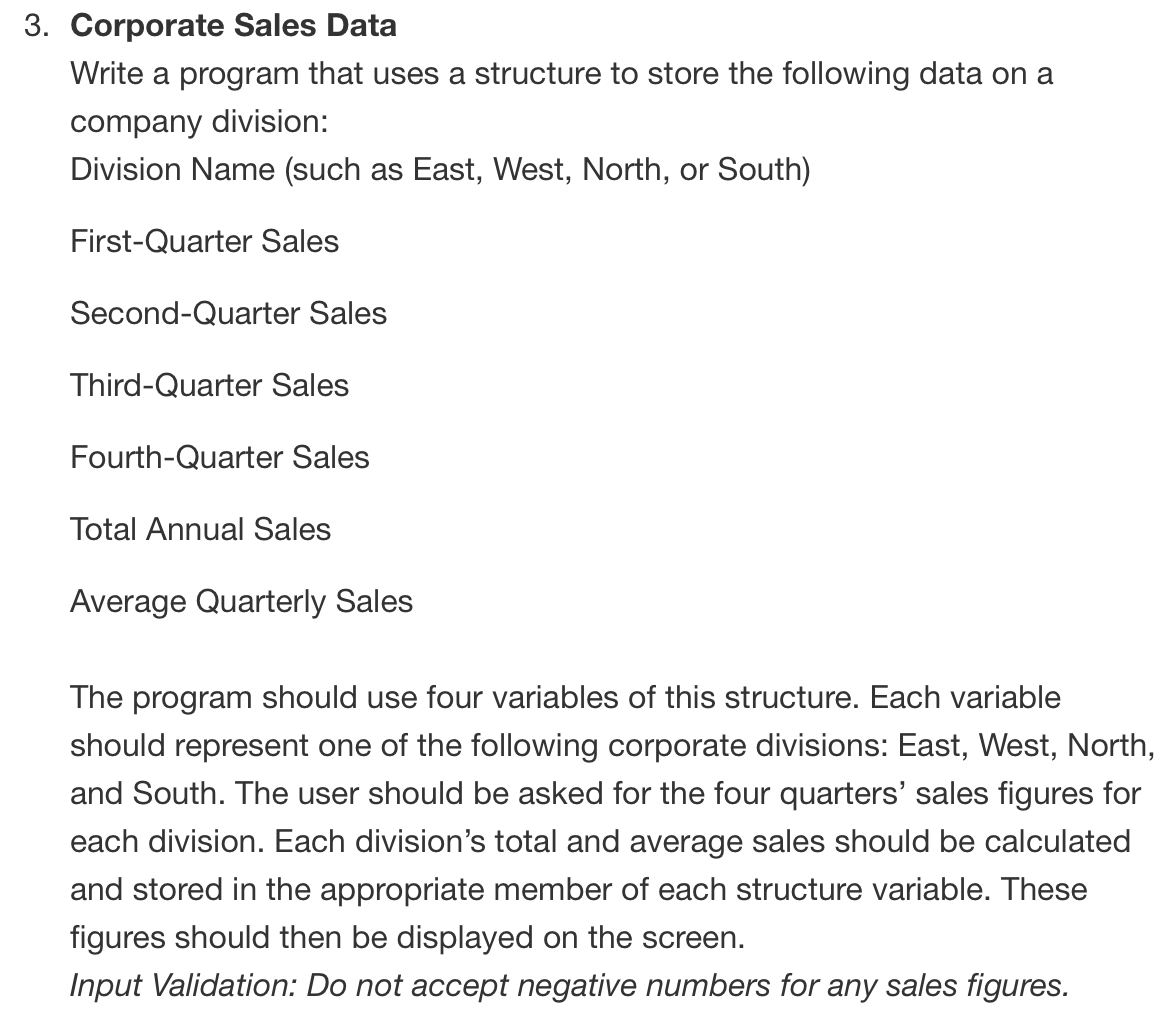
***COMSC165 Lab-3 Due: 7/08/18 11:59 PM***

Follow the lab format under the general folder in Canvas to complete this lab.



#ifndef CORPORATESALES\_H

#define CORPORATESALES\_H

#include <string>

class CorporateSales

{

public:

// structure with corporate divisions attributes

struct DivisionSales{

std::string divisionName;

double quarterOne;

double quarterTwo;

double quarterThree;

double quarterFour;

double annualSales;

double averageQuarterlySales;

};

};

#endif // CORPORATESALES\_H

#include "CorporateSales.h"

#include <iostream>

#include <string>

using namespace std;

// create an object of the structure and return it

void CorporateSales::createDivisionAndReadDataAndDisplayData(string name){

DivisionSales divSalesObject;

divSalesObject.divisionName = name;

divSalesObject.quarterOne = 0;

divSalesObject.quarterTwo = 0;

divSalesObject.quarterThree = 0;

divSalesObject.quarterFour = 0;

divSalesObject.annualSales = 0;

divSalesObject.averageQuarterlySales = 0;

cout << "Enter the " << name << "'s quarterly sales." << endl;

cout << "Quarter 1: ";

cin >> divSalesObject.quarterOne;

while (divSalesObject.quarterOne < 0){

cout << "ERROR : You must enter a positive amount of sales.\n" << endl;

cout << "Re-enter the sales for the division: ";

cin >> divSalesObject.quarterOne;

cout << "\n";

}

cout << "Quarter 2: ";

cin >> divSalesObject.quarterTwo;

while (divSalesObject.quarterTwo < 0){

cout << "ERROR : You must enter a positive amount of sales.\n" << endl;

cout << "Re-enter the sales for the division: ";

cin >> divSalesObject.quarterTwo;

cout << "\n";

}

cout << "Quarter 3: ";

cin >> divSalesObject.quarterThree;

while (divSalesObject.quarterThree < 0){

cout << "ERROR : You must enter a positive amount of sales.\n" << endl;

cout << "Re-enter the sales for the division: ";

cin >> divSalesObject.quarterThree;

cout << "\n";

}

cout << "Quarter 4: ";

cin >> divSalesObject.quarterFour;

while (divSalesObject.quarterFour < 0){

cout << "ERROR : You must enter a positive amount of sales.\n" << endl;

cout << "Re-enter the sales for the division: ";

cin >> divSalesObject.quarterFour;

cout << "\n";

}

cout << "Debug for annual sales: ";

divSalesObject.annualSales += divSalesObject.quarterOne;

divSalesObject.annualSales += divSalesObject.quarterTwo;

divSalesObject.annualSales += divSalesObject.quarterThree;

divSalesObject.annualSales += divSalesObject.quarterFour;

cout << divSalesObject.annualSales << endl;

divSalesObject.averageQuarterlySales = (divSalesObject.annualSales / 4);

cout << "Debug for average sales: ";

cout << divSalesObject.averageQuarterlySales << endl << endl;

}

#include <iostream>

#include "CorporateSales.h"

#include <string>

using namespace std;

int main(){

string east = "East";

string west = "West";

string north = "North";

string south = "South";

CorporateSales cso;

cso.createDivisionAndReadDataAndDisplayData(east);

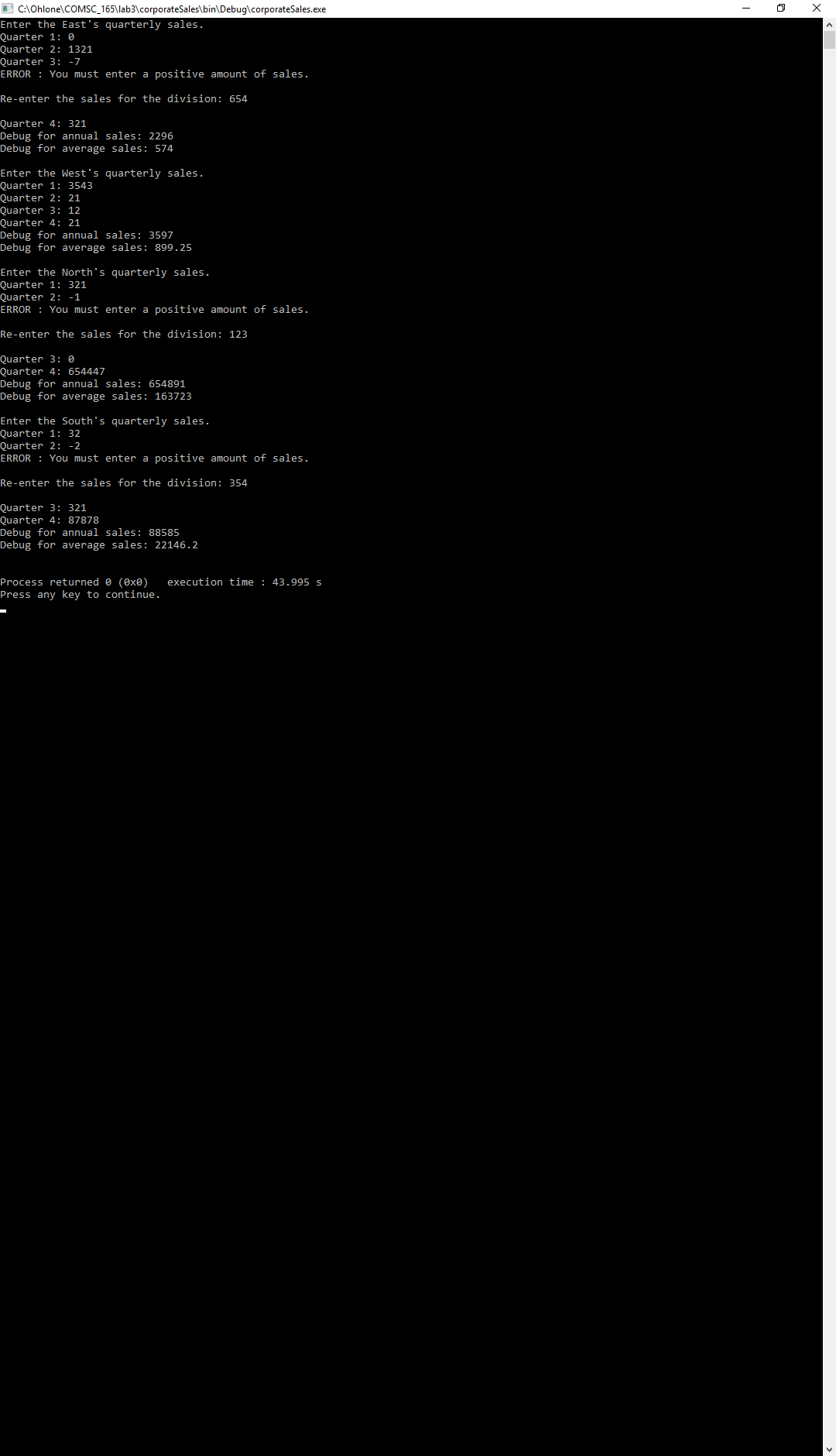
cso.createDivisionAndReadDataAndDisplayData(west);

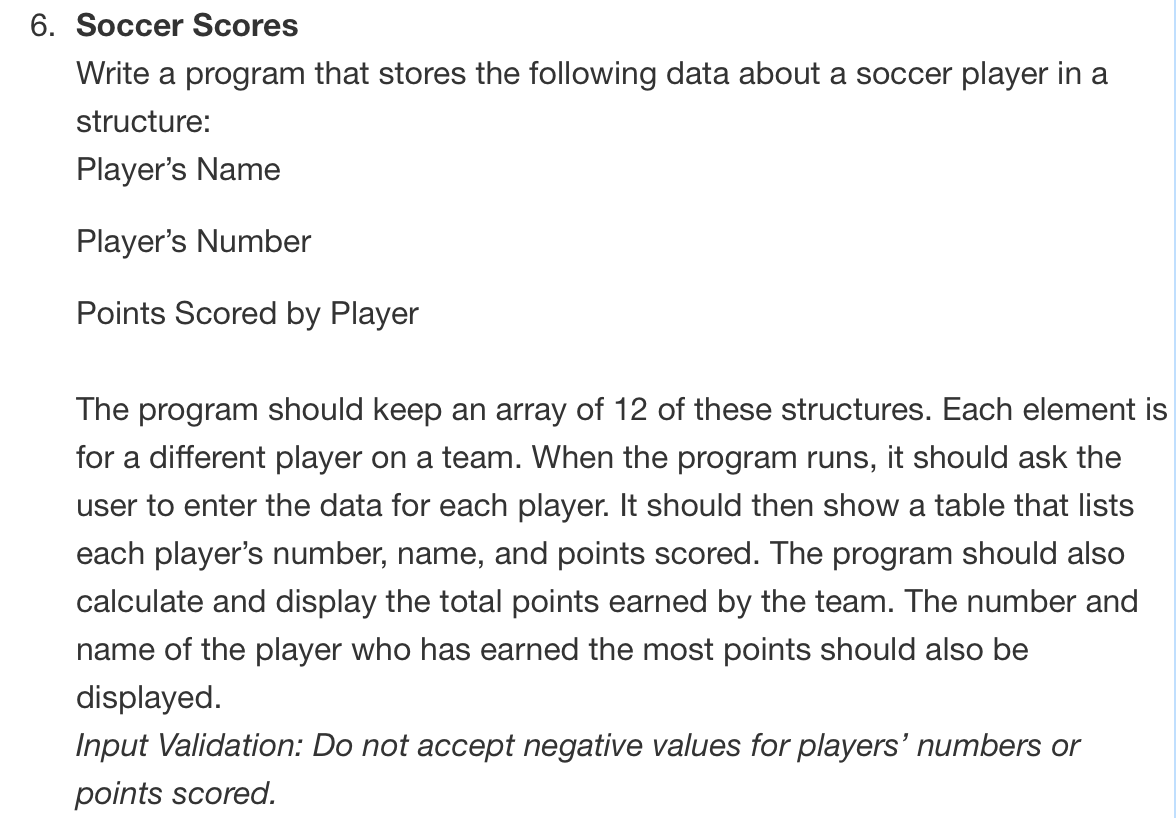
cso.createDivisionAndReadDataAndDisplayData(north);

cso.createDivisionAndReadDataAndDisplayData(south);

return 0;

}





#ifndef SOCCERSCORES\_H

#define SOCCERSCORES\_H

#include <string>

class SoccerScores

{

public:

struct SoccerPlayer{

std::string name;

int jerseyNumber;

int goalsScored;

};

int teamSize;

SoccerPlayer\* soccerTeam;

SoccerScores();

void getPlayerData(SoccerPlayer [], int);

void calculateAndDisplayTeamData(SoccerPlayer [], int);

};

#endif // SOCCERSCORES\_H

#include "SoccerScores.h"

#include <iostream>

#include <iomanip>

#include <string>

using namespace std;

// constructor initializes team size to 12

// creates array of soccerPlayer with 12 elements

SoccerScores::SoccerScores(){

teamSize = 12;

soccerTeam = new SoccerPlayer[teamSize];

}

// read user input for soccer play data

void SoccerScores::getPlayerData(SoccerPlayer team[], int SIZE){

cout << "Enter the name for each player on the team "

<< "along with their number and points scored."

<< endl;

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

cout << "\nPlayer " << (i+1) << endl;

cout << "Name: ";

cin >> team[i].name;

cout << "Number: ";

cin >> team[i].jerseyNumber;

while (team[i].jerseyNumber < 0){

cout << "ERROR : You must enter a positive number." << endl;

cout << "Re-enter a valid number: ";

cin >> team[i].jerseyNumber;

cout << "\n";

}

cout << "Goals Scored: ";

cin >> team[i].goalsScored;

while (team[i].goalsScored < 0){

cout << "ERROR : You must enter a positive number." << endl;

cout << "Re-enter a valid number: ";

cin >> team[i].goalsScored;

}

}

}

// print out a table that displays

// player names, numbers, points scored

void SoccerScores::calculateAndDisplayTeamData(SoccerPlayer team[], int SIZE){

cout << "\nPlayer Name\tJersey Number\tGoals Scored" << endl;

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

cout << team[i].name

<< "\t\t"

<< team[i].jerseyNumber

<< "\t\t"

<< team[i].goalsScored

<< endl;

}

int mostGoalsScoredIndex = 0;

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

if (team[i].goalsScored > mostGoalsScoredIndex){

mostGoalsScoredIndex = i;

}

}

cout << "\n"

<< team[mostGoalsScoredIndex].name

<< " #"

<< team[mostGoalsScoredIndex].jerseyNumber

<< " scored the most goals on the team." << endl;

// print out player with most points scored

}

#include <iostream>

#include "SoccerScores.h"

#include <iomanip>

#include <string>

using namespace std;

int main(){

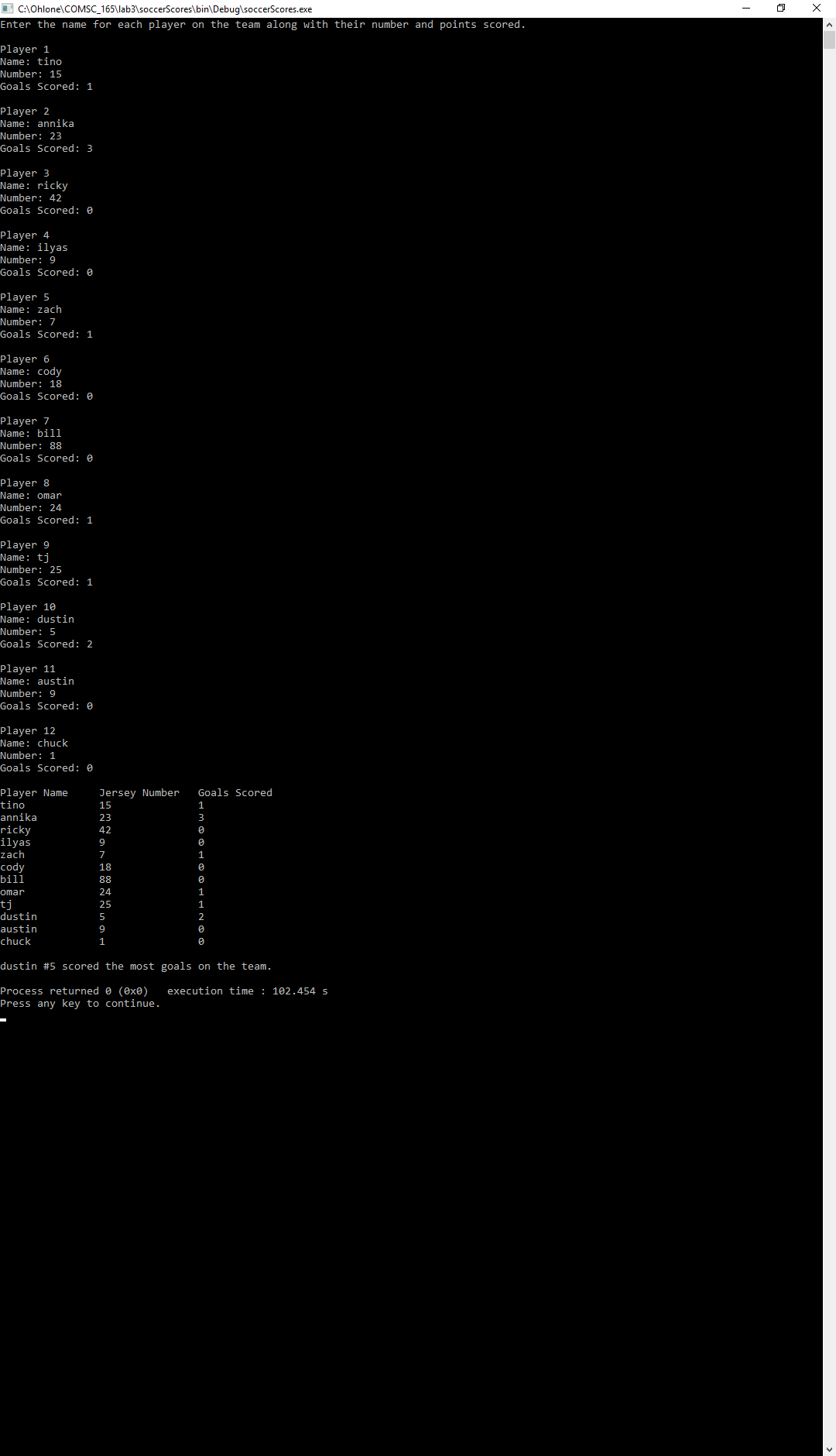
SoccerScores sco;

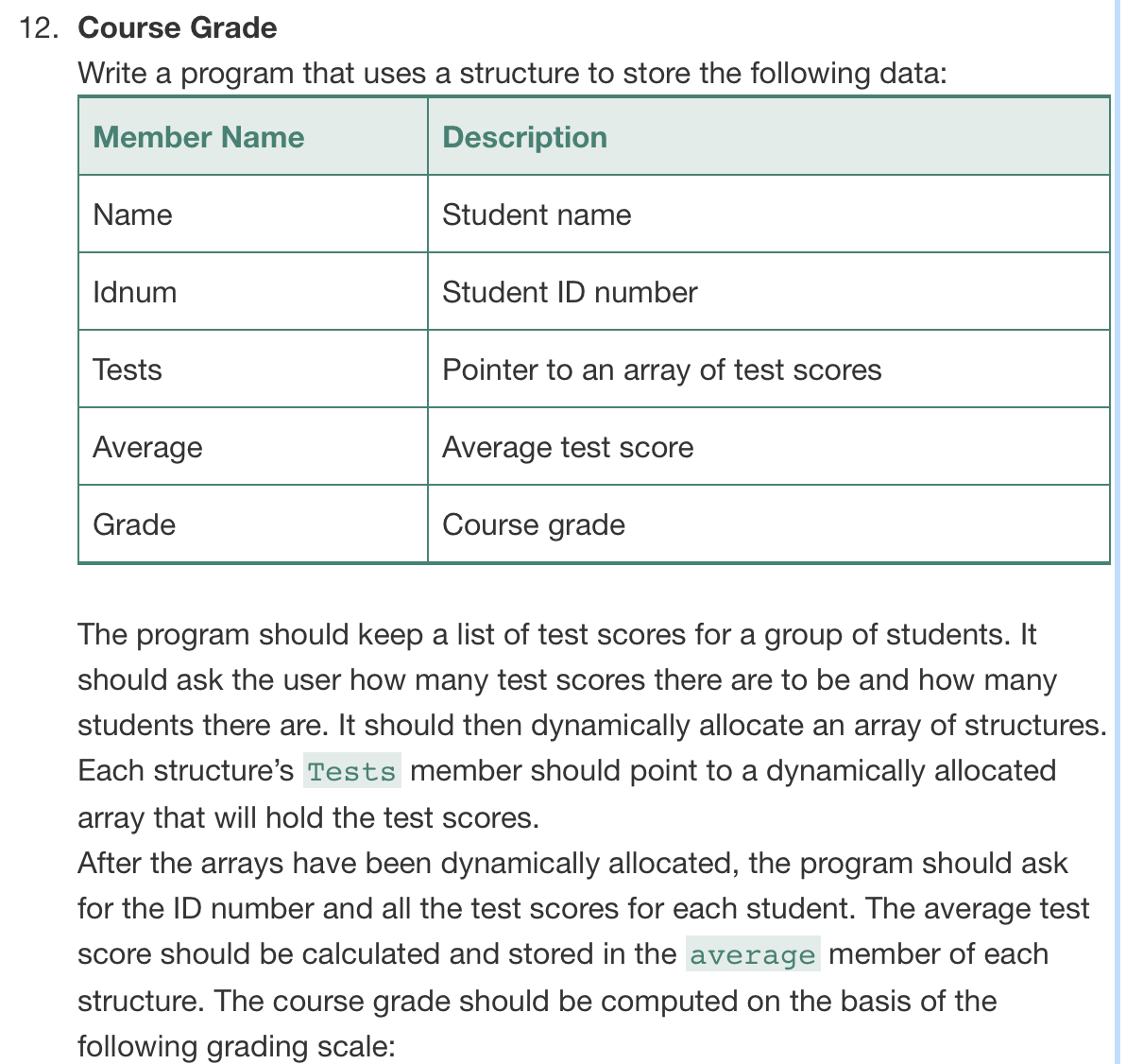
sco.getPlayerData(sco.soccerTeam, sco.teamSize);

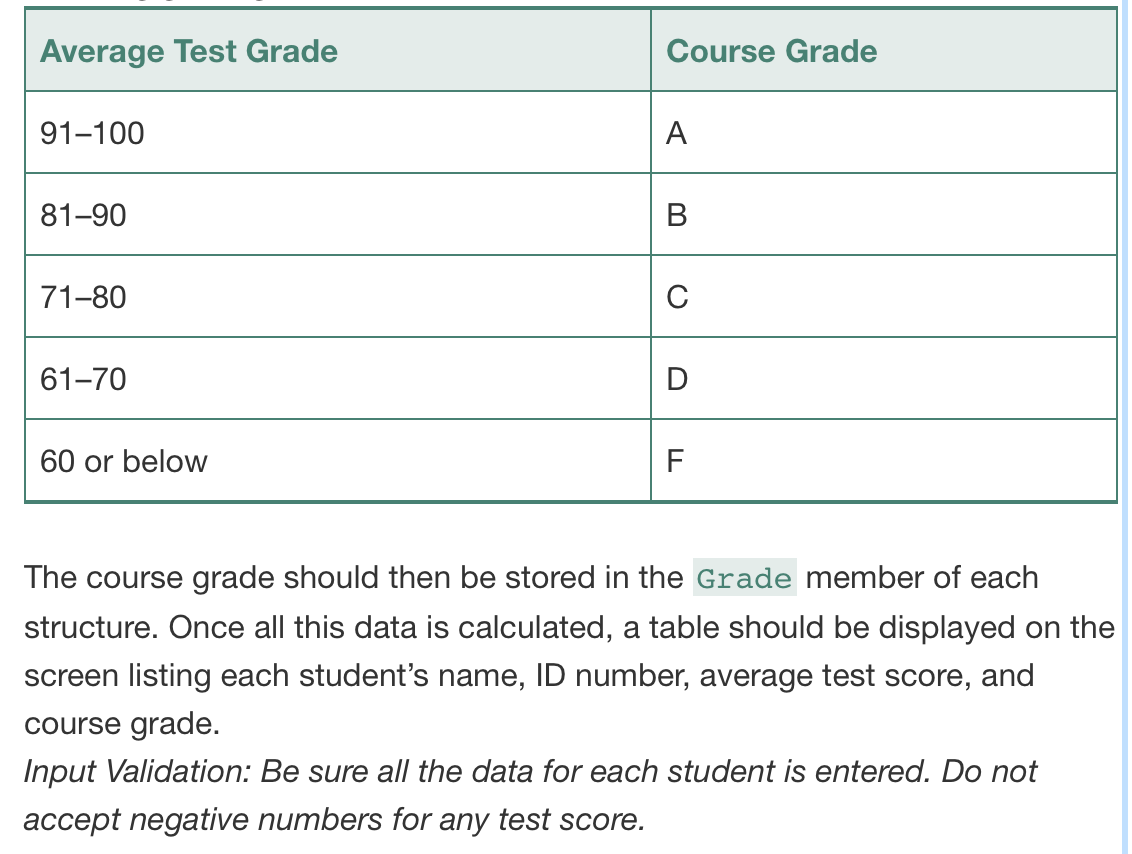
sco.calculateAndDisplayTeamData(sco.soccerTeam, sco.teamSize);

return 0;

}







#ifndef COURSEGRADE\_H

#define COURSEGRADE\_H

#include <string>

// global structure of a student with attributes

struct Student{

std::string name; // student name

int idNumber; // student id number

int \*tests; // pointer to a dynamic array of test scores

double average; // average test score

char grade; // course grade

};

class CourseGrade{

public:

int numberOfStudents;

int numberOfTests;

Student \*studentGroup = nullptr;

// read user input for # of students

void setNumberOfStudents();

// read user input for # of tests

void setNumberOfTests();

// dynamically allocate an array of student struct

void setStudentArray(int);

// dynamically allocate array for test scores

void createTestArray(Student [], int, int);

// ask user for student id number and test scores for each student

// validate negative test scores

void setStudentData(Student [], int, int);

// calculate average test scores and store in struct average attribute

void calculateAverageTestScores(Student [], int, int);

// calculate final grade and store in struct grade attribute

void setStudentGrade(Student [], int);

// display student info in a table

void displayStudentData(Student [], int);

// delete dynamic array

void deleteDynArray(Student []);

};

#endif // COURSEGRADE\_H

#include "CourseGrade.h"

#include <iostream>

#include <string>

using namespace std;

// read user input for # of students

void CourseGrade::setNumberOfStudents(){

cout << "How many students are in the class? ";

cin >> numberOfStudents;

while (numberOfStudents <= 0){

cout << "ERROR : You must enter a positive number." << endl;

cout << "Re-enter a valid number: ";

cin >> numberOfStudents;

}

}

// read user input for # of tests

void CourseGrade::setNumberOfTests(){

if (numberOfStudents == 1){

cout << "How many tests did that student take? ";

cin >> numberOfTests;

while (numberOfTests <= 0){

cout << "ERROR: Invalid number." << endl;

cout << "Re-enter a valid number: ";

cin >> numberOfTests;

}

}else{

cout << "How many tests did each student take? ";

cin >> numberOfTests;

while (numberOfTests <= 0){

cout << "ERROR: Invalid number." << endl;

cout << "Re-enter a valid number: ";

cin >> numberOfTests;

}

}

}

// dynamically allocate an array of student struct

void CourseGrade::setStudentArray(int arrSize){

studentGroup = new Student[arrSize];

}

void CourseGrade::createTestArray(Student arr[], int numStudents, int numTests){

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

arr[i].tests = new int[numTests];

}

}

// ask user for student id number and test scores for each student

// validate negative test scores

void CourseGrade::setStudentData(Student arr[], int numStudents, int numTests){

cout << "\nEnter each student's name, id number, and test score." << endl;

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

cout << "Student "

<< (i+1)

<< "'s' name: ";

cin >> arr[i].name;

cout << arr[i].name

<< "'s id number: ";

cin >> arr[i].idNumber;

for (int j = 0; j < numTests; ++j){

cout << arr[i].name

<< "'s test"

<< (j+1)

<< ": ";

cin >> arr[i].tests[j];

while (arr[i].tests[j] < 0){

cout << "ERROR : You must enter a positive number." << endl;

cout << "Re-enter a valid number: ";

cin >> arr[i].tests[j];

}

}

cout << endl;

}

}

// calculate average test scores and store it in struct's average attribute

void CourseGrade::calculateAverageTestScores(Student arr[], int numStudents, int numTests){

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

for (int j = 0; j < numTests; ++j){

arr[i].average += arr[i].tests[j];

}

arr[i].average = arr[i].average / numberOfTests;

}

}

// calculate final grade and store in struct's grade attribute

void CourseGrade::setStudentGrade(Student arr[], int numStudents){

int maxTestScore = 100;

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

if(arr[i].average >= 90 && arr[i].average <= maxTestScore){

arr[i].grade = 'A';

}else if(arr[i].average >= 80 && arr[i].average < 90){

arr[i].grade = 'B';

}else if(arr[i].average >= 70 && arr[i].average < 80){

arr[i].grade = 'C';

}else if(arr[i].average >= 60 && arr[i].average < 70){

arr[i].grade = 'D';

}else{

arr[i].grade = 'F';

}

}

}

// display student data info in a table

void CourseGrade::displayStudentData(Student arr[], int numStudents){

cout << "\nStudent Name\tID Number\tAverage Test Score\tCourseGrade" << endl;

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

cout << arr[i].name

<< "\t\t"

<< arr[i].idNumber

<< "\t\t"

<< arr[i].average

<< "\t\t\t"

<< arr[i].grade

<< endl;

}

}

// delete dynamic array

void CourseGrade::deleteDynArray(Student arr[]){

delete [] arr;

}

#include "CourseGrade.h"

#include <iostream>

#include <string>

using namespace std;

int main(){

CourseGrade cgo;

cgo.setNumberOfStudents();

cgo.setNumberOfTests();

cgo.setStudentArray(cgo.numberOfStudents);

cgo.createTestArray(cgo.studentGroup, cgo.numberOfStudents, cgo.numberOfTests);

cgo.setStudentData(cgo.studentGroup, cgo.numberOfStudents, cgo.numberOfTests);

cgo.calculateAverageTestScores(cgo.studentGroup, cgo.numberOfStudents, cgo.numberOfTests);

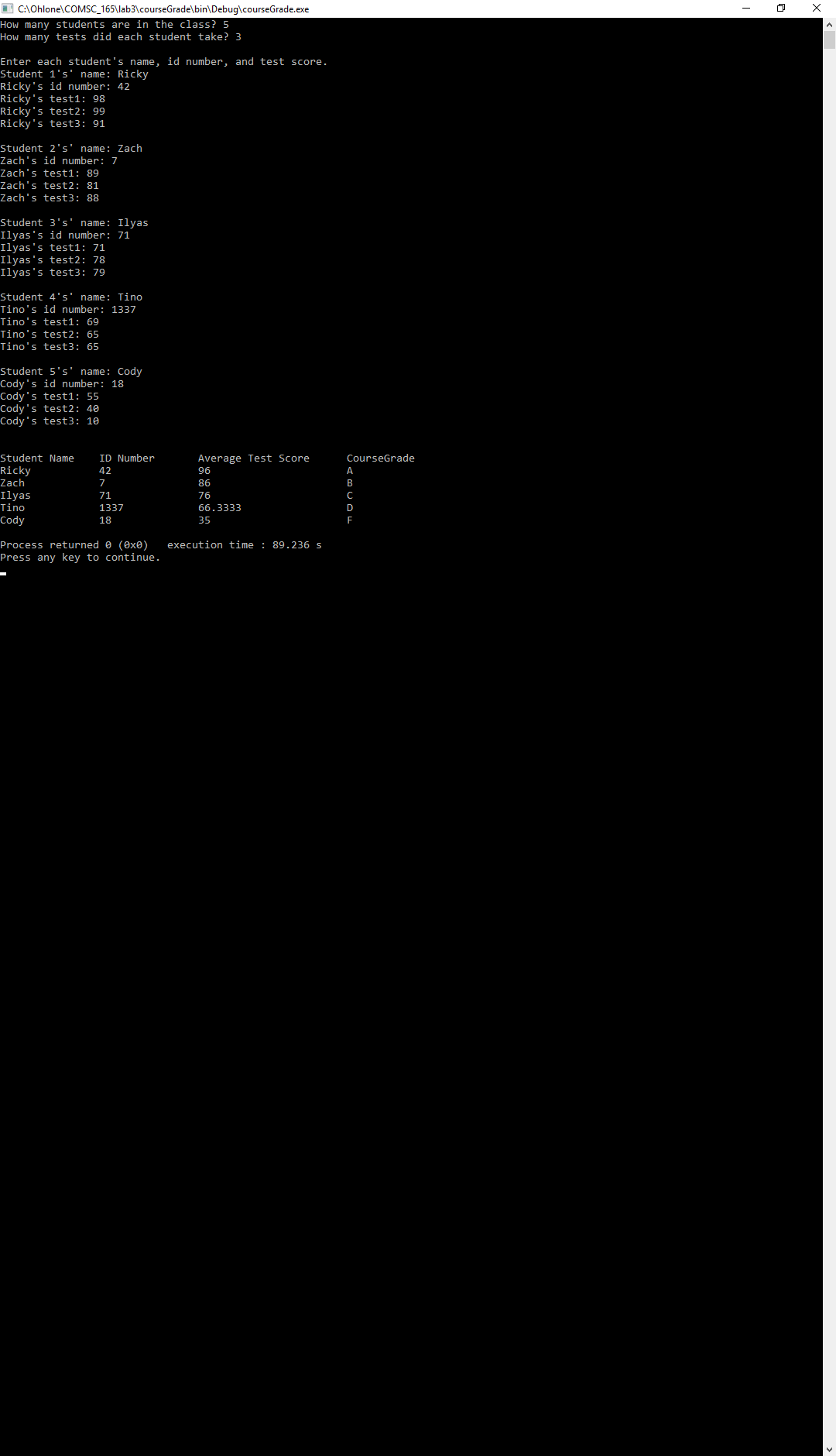
cgo.setStudentGrade(cgo.studentGroup, cgo.numberOfStudents);

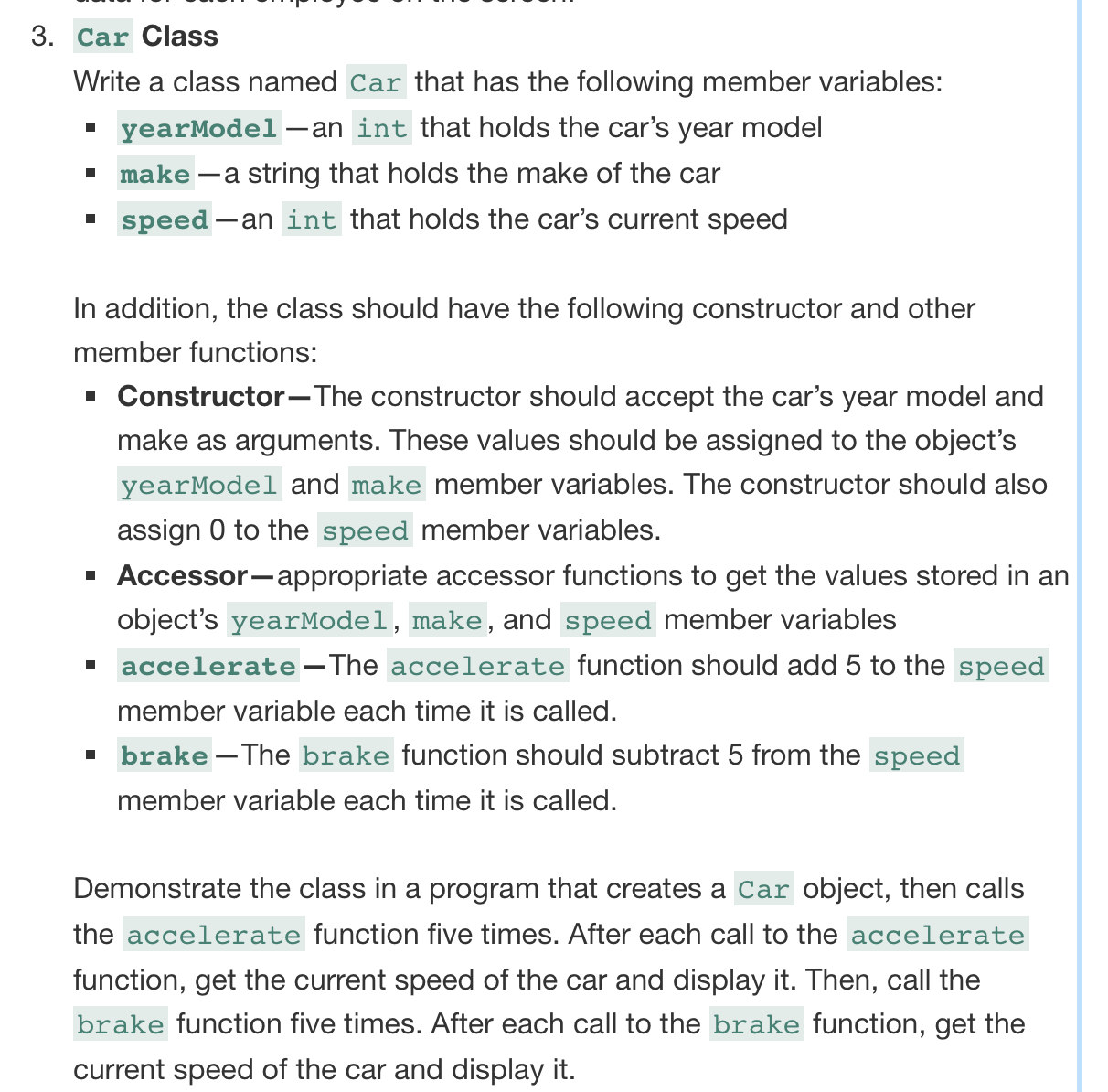
cgo.displayStudentData(cgo.studentGroup, cgo.numberOfStudents);

cgo.deleteDynArray(cgo.studentGroup);

return 0;

}





#ifndef CAR\_H

#define CAR\_H

#include <string>

class Car

{

public:

/\* constructor that takes yearModel and make as arguments

\* and assigns them to objects yearModel and make members

\* and assigns 0 to the member speed

\*/

Car(int, std::string);

// get value stored in yearModel member

int getYearModel() const;

// get value stored in speed member

int getSpeed() const;

// get value stored in make member

std::string getMake() const;

// adds 5 to the speed member variable

void accelerate(int);

// subtracts 5 from the speed member variable

void brake(int);

private:

int yearModel;

int speed;

std::string make;

};

#endif // CAR\_H

#include "Car.h"

#include <iostream>

#include <string>

using namespace std;

/\*

\* constructor that takes yearModel and make as arguments

\* and assigns them to objects yearModel and make members

\* and assigns 0 to the member speed

\*/

Car::Car(int model, string carMake){

yearModel = model;

make = carMake;

speed = 0;

}

/\*

\* accessor for yearModel to get the value stored

\* in an objects yearModel member variable

\*/

int Car::getYearModel() const {

return yearModel;

}

/\*

\* accessor for speed to get the value stored

\* in an objects speed member variable

\*/

int Car::getSpeed() const {

return speed;

}

/\*

\* accessor for make to get the value stored

\* in an objects make member variable

\*/

string Car::getMake() const {

return make;

}

/\*

\* the accelerate function adds 5 to the speed

\* member variable each time it is called

\*/

void Car::accelerate(int){

speed = speed + 5;

}

/\*

\* the brake function subtracts 5 from the

\* speed member variable each time it's called

\*/

void Car::brake(int){

speed = speed - 5;

}

#include "Car.h"

#include <iostream>

#include <string>

using namespace std;

int main(){

int tempYearModel;

string tempMake;

// get the make of the car

cout << "What is the make of your car? ";

getline(cin, tempMake);

// get the year of the car

cout << "What is the year of your car? ";

cin >> tempYearModel;

// create car object

Car co(tempYearModel, tempMake);

for (int i = 0; i < 5; ++i){

int tempSpeed = co.getSpeed();

co.accelerate(co.getSpeed());

cout << "\nThe speed of the car is "

<< tempSpeed

<< " mph.";

}

for (int j = 0; j <= 5; ++j){

int tempSpeed = co.getSpeed();

co.brake(co.getSpeed());

cout << "\nThe speed of the car is "

<< tempSpeed

<< " mph.";

}

return 0;

}

