```cpp
// Ch 7_GradeBook.cpp : Defines the entry point for the console application.

#include "stdafx.h"
#include <iostream>
#include <array>
#include "GradeBook.h"
using namespace std;

int main() {
	// array of student grades
	const array<int, GradeBook::students> grades{
		87, 68, 94, 100, 83, 78, 85, 91, 76, 87 };
	string courseName{ "CS107 C++ Programming" };

	GradeBook cppGradeBook(courseName, grades);
	cppGradeBook.displayMessage();
	cppGradeBook.processGrades();

	system("pause");
	return 0;
}

==========================================================================

// definition of class GradeBook that uses an array to store test gades
#include <string>
#include <array>
#include <iostream>
#include <iomanip>

class GradeBook {
public:
	// constant number of students who took the test
	static const size_t students{ 10 };

	//constructor initializes courseName and grades array
	GradeBook(const std::string& name,
		const std::array<int, students>& gradesArray)
		: courseName{ name }, grades{ gradesArray } {
}

// function to set the course name
void setCourseName(const std::string& name) {
	courseName = name; // store the course name
}

// function to get the course name
const std::string& getCourseName() const {
	return courseName;
}

// display a welcome message to the GradeBook user
void displayMessage() const {
	// call getCourseName to get the name of this GradeBook's course
	std::cout << "Welcome to the grade book for\n" << getCourseName()
		<< "!" << std::endl;
}

// perform various operations on the data (none modify the data)
void processGrades() const {
	outputGrades(); // output grades array

	// call function getAverage to calculate the average grade
	std::cout << std::setprecision(2) << std::fixed;
	std::cout << "\nClass average is " << getAverage() << std::endl;

	// call functions getMinimum and getMaximum
	std::cout << "Lowest grade is: " << getMinimum()
		<< "\nHighest grade is " << getMaximum() << std::endl;

	outputBarChart(); // display grade distribution chart
}

// function to find minimum grade
int getMinimum() const {
	int lowGrade{ 100 };

	// loop through grades array
		for (int grade : grades) {
			// if current grade is less than low grade assign it to
			// lowGrade
			if (grade < lowGrade) {
				lowGrade = grade; // new lowest grade
			}
		}

	return lowGrade;
}

// function to find the highest grade
int getMaximum() const {
	int highGrade{ 0 };

	// loop through grades array


	for (int grade : grades) {
		// if current grade is higher than highGrade assign it to
		// highGrade
		if (grade > highGrade) {
			highGrade = grade;
		}

		return highGrade;
	}
}

// determain average grade for tests
double getAverage()    const {
	int total{ 0 };

	// sum grades in array
	for (int grade : grades) {
		total += grade;
	}

	// return average of grades
	return static_cast<double>(total) / grades.size();
}

// output bar chart displaying grade distribution
void outputBarChart() const {
	std::cout << "\nGrade distribution: " << std::endl;

	// stores frequencies of in each range of 10 grades
	const size_t frequencySize{ 11 };
	std::array<unsigned int, frequencySize> frequency{}; // init to 0s

	// for each grade, increment the appropriate frequency
	for (int grade : grades) {
		++frequency[grade / 10];
	}

	// for each grade frequency, print bar in chart
	for (size_t count{ 0 }; count < frequencySize; ++count) {
		// output bar labels ("0-9:",... "90-99", "100:")
		if (0 == count) {
			std::cout << " 0-9: ";
		}
		else if (10 == count) {
			std::cout << " 100: ";
		}
		else {
			std::cout << count * 10 << "-" << (count * 10) + 9 << ": ";
		}

		// print bar of asterisks
		for (unsigned int stars{ 0 }; stars < frequency[count]; ++stars) {
			std::cout << "*";
		}

		std::cout << std::endl;
	}
}

	// output the contents of the grades array
	void outputGrades() const {
		std::cout << "\nThe grades are:\n\n";

		// output each students grade
		for (size_t student{ 0 }; student < grades.size(); ++student) {
			std::cout << "Student " << std::setw(2) << student + 1 << ": "
				<< std::setw(3) << grades[student] << std::endl;
		}
	}

private:
	std::string courseName; // course name for this gradebook
	std::array<int, students> grades; // array of student grades
};
```