Hints for the C++ Revel Quizzes and Programming Projects

For Introduction to C++ and Data Structures 4E by Y. Daniel Liang

If you see any errors or have comments or suggestions, please email me at y.daniel.liang@gmail.com.

Please check this page often. We update this page frequently in response to student questions. Last update: 02/08/2021.

NOTE: The quiz is a third-party product. Many end-of-section quizzes use a different naming convention for variables and functions. For example, it uses `number_of_credits` to name a variable and `get_radius()` to name a getter function. But it would be named `numberOfCredits` and `getRadius()` if the naming convention of this book is used.

NOTE: For all quizzes and programming projects, use the double type rather than the float type for real numbers, because some quizzes may not grade correctly if the float type is used.

Hints and Errata for End-of-Section Assignments (Programming Quizzes):

Chapter 2 Quiz 2.7 #1 and #2:
Note: Revel assignable exercises is a third-party product. The naming of the constants in these quizzes violates our naming convention for constants.

Chapter 3 Quiz 3.4 #3:
Note: if `callsReceived < 0` or `operatorsOnCall <= 0`, print the message `"INVALID"`. You can write it using the following template:

```
cin >> callsReceived;
cin >> operatorsOnCall;
if (operatorsOnCall <= 0)
  // Write the necessary code here
else if (callsReceived < 0)
  // Write the necessary code here
else
  // Write the necessary code here
```

After you learned Boolean operators in Section 3.10, you can write a simpler statement as follows:

```
cin >> callsReceived;
cin >> operatorsOnCall;
if (operatorsOnCall <= 0 || callsReceived < 0)
  // Write the necessary code here
else
  // Write the necessary code here
```

Chapter 7 Quiz 7.11 #1:
Note: Change the description to:
Declare a `char` array named `line` of size 50, and write a statement that reads in the next line of console input into this array. (Assume the line may contain whitespace characters and the total number of the characters in the line is less than 50.)

Chapter 8 Quiz 8.3 #2:
Please consider the case that all the numbers in the matrix may be negative when writing your code.

Chapter 9 Quiz 9.2 #1, #2, #3:

Note: See the small text in Figure 9.2 on how to declare objects using constructors.

Chapter 9 Quiz 9.9 #3:
Note: This exercise needs to declare static members. However, static members are introduced in Section 10.5. This mistake will be fixed in the future edition of this book.

Chapter 9 Quiz 9.9 #4:
Note: This exercise uses static members. However, static members are introduced in Section 10.5. This mistake will be fixed in the future edition of this book. You need to initialize static data member nCounters as

```
int Counter::nCounters = 0;
```

This line has to be put as the first line in your code. Don't forget this line. The static variable is not initialized in the header file.

Chapter 10 Quiz 10.5 #2:
Note: Don't use the #include "Counter.h", because REVEL assumes that your Counter header is in the same file with the implementation. You need to initialize static data member nCounters as

```
int Counter::nCounters = 0;
```

Chapter 11 Quiz 11.12 #1:
Note: Don't use the **Window::** qualification in your code, because REVEL assumes that your code will be inserted directly in the Window class definition as an inline implementation.

Chapter 11 Quiz 11.15 #1:
Note: Don't use the **GraphicProgram::** qualification in your code, because REVEL assumes that your code will be inserted directly in the GraphicProgram class definition as an inline implementation.

Chapter 15 Quiz 15.2 #1:
Note: You need to define the SavingsAccount class and also implement the SavingsAccount's constructor and the getInterestRate and getInterestType functions.

Chapter 15 Quiz 15.2 #2:
Note: You need to define the WindowWithBorder class and also implement the WindowWithBorder's constructor and the getUseableWidth function.

Chapter 15 Quiz 15.2 #3:
Note: You need to define the CameraPhone class and also implement the CameaPhone's constructor and the numPictures function.

Hints for End-of-Chapter Assignments (Programming Projects):

Chapter 1: Programming Project 1: Exercise01_01
Hint:
Please check your spelling of the words carefully. It is case sensitive. The words are separated by exactly one space.

Chapter 1: Programming Project 3: Exercise01_11
Hint:
In one year, the population will be
312032486 + 365 * 24 * 60 * 60 / 7.0 - 365 * 24 * 60 * 60 / 13.0 + 365 * 24 * 60 * 60 / 45.0
In two years, the population will be
312032486 + 2 * 365 * 24 * 60 * 60 / 7.0 - 2 * 365 * 24 * 60 * 60 / 13.0 + 2 * 365 * 24 * 60 * 60 / 45.0
You can compute the population in three years, four years, and five years.

Chapter 2: Programming Project 1: Exercise02_05

How would you write this program? Here are some hints:
Step 1: Prompt the user to enter subtotal (double) and gratuity rate into variables subtotal (double) and gratuityRate (double). Note the gratuityRate is in percent. For example, if it is 15%, you enter 15 as the input for gratuityRate.
Step 2: Compute gratuity: gratuity = subtotal * gratuityRate / 100.
Step 3: Compute total: total = subtotal + gratuity.
Step 4: Display gratuity and total. Note you need to first display gratuity and then total. For all the programming projects, please pay attention to output order.

Chapter 2: Programming Project 2: Exercise02_07
Hint: You must use the int type for all variables in this program. Please note that the % operator only works for integers in C++. See LiveExample 2.7 in Section 2.8.3 for reference.
Now, the key to solve this problem is to use the correct math.
Step 1: Get the totalNumberOfDays from the minutes (i.e., totalNumberOfDays = minutes / (24 * 60)).
Step 2: Get the numberOfYears from the numberOfDays (i.e., numberOfYears = numberOfDays / 365).
Step 3: Get the remainingNumberOfDays from totalNumberOfDays (i.e., remainingNumberOfDays = totalNumberOfDays % 365).

Chapter 2: Programming Project 3: Exercise02_15
How would you write this program? Here are some hints:
Step 1: Prompt the user to enter two points. Each point has two values for x- and y-coordinates (double).
Step 2: Compute the distance using the distance formula given in the problem description. The correct formula for computing the distance is pow((x2 – x1) * (x2 – x1) + (y2 – y1) * (y2 – y1), 0.5)
Step 3: Display the result.

Chapter 2: Programming Project 5: Exercise02_23
Hint: The annualInterestRate (double) is entered in percentage. In the formula, you need to use monthlyInterestRate (double), which is annualInterestRate / 1200. See LiveExample 2.11 in Section 2.14 for reference.

Chapter 3: Programming Project 1: Exercise03_01
How would you write this program? Here are some hints:
Step 1: Prompt the user to enter a, b, c. The numbers are double.
Step 2: Compute discriminant.
Step 3: if (discriminant < 0), display an error message.
Step 4: else if (discriminant == 0), compute and display one root.
Step 5: else compute and display two roots.

Common errors from this project: In Step 1, your code should read in float values. In Step 2, discriminant is discriminant = b * b – 4 * a * c. In Step 5, please note that root1 is r1 = (-b + discriminant ** 0.5) / (2 * a). Another common mistake is to compute the roots before the if statement. Please compute roots in Step 5.

Chapter 3: Programming Project 2: Exercise03_03
How would you write this program? Here are some hints:
Step 1: Prompt the user to enter a, b, c, d, e, f. The numbers are double.
Step 2: Compute detA = a * d – b * c.
Step 3: if (detA == 0), display an error message.
Step 4: else compute x and y, and display the result.

Common errors from this project: A common mistake is to compute x and y before the if statement. Please compute x and y in Step 4.

Chapter 3: Programming Project 3: Exercise03_09
Prompt the user to enter year (int) and test if year is a leap year.

Hint: To test if a year is a leap year, see Section 3.11.

Chapter 3: Programming Project 4: Exercise03_33
Hint: Prompt the user to enter year, month, and dayOfMonth. Note that all input are integers. You need to adjust year and month as follows:
if month is 1, set month = 13 and year = year – 1.
if month is 2, set month = 14 and year = year – 1.
You can now use Zeller's formula to compute h:

$$h = \left(q + \frac{26(m+1)}{10} + k + \frac{k}{4} + \frac{j}{4} + 5j \right) \% \, 7$$

Chapter 3: Programming Project 5: Exercise03_19
All the input are double.
Hint: A point (x, y) is in the circle centered at (0, 0) with radius 10 if the distance to the center is <= 10.

Chapter 4: Programming Project 3: Exercise04_13
How would you write this program? Here are some hints:
Step 1: Prompt the user to enter a character for a hex digit into char hex.
Step 2: Use a multi-way if statement or a switch statement to obtain the corresponding binary string based on the hex value.
Step 3: Display the result.

Chapter 4: Programming Project 4: Exercise04_19
When you display the result, print the three cities in one line, separated by exactly one space.

Chapter 4: Programming Project 5: Exercise04_23
Hint: You will need to use s.length() and isdigit(ch) function in the program.

Chapter 5: Programming Project 1: Exercise05_01
How would you write this program? Here are some hints:
Step 1: Declare and initialize variables. Your program needs to count the number of positives, number of negatives, and total. Think about what initial values are appropriate for these variables. Note that the average can be computed using total / (numberOfPositives + numberOfNegatives).
Step 2: Prompt the user to read a number into variable number.
Step 3: Write a while loop to do the following:
Step 3.1: the loop continuation condition is (number =! 0)
Step 3.2: If number > 0, increase numberOfPositives by 1; if number < 0, increase numberOfNegatives by 1.
Step 3.3: Add number to total.
Step 3.4: Prompt the user to a new input and assign it to number.
Step 4: After the loop is finished, if (numberOfPositives + numberOfNegatives == 0), this means "No numbers are entered except 0". Your program should display exactly this message. Otherwise, display the number of positives, negatives, and average. To compute average, use 1.0 * total / (numberOfPositives + numberOfNegatives).

Chapter 5: Programming Project 2: Exercise05_09
How would you write this program? Here are some hints:
First, I suggest that you to read Section 5.11.2. This section will give you half of the solution.
Step 1: Note that the current year tuition is 10000. In one year (same as to say after the first year), the tuition will be tuition += tuition * 1.05 (10500). Compute the tuition in 10 years using a loop and save the result in variable tuition. This will be the tuition in 10 years. Check your result. It should be 16288.95.
Step 2: Now compute the tuition after the 11^{th} year, 12^{th} year, 13^{th} year. The tuition after the 10^{th} year is 16288.95. The sum of the tuition after the 10^{th}, 11^{th}, 12^{th}, and 13^{th} is the total cost of four years tuition in ten years (same as to say after the 10^{th} year).

Chapter 5: Programming Project 3: Exercise05_11
Note that we assume that the number of students is at least 2. So, your program does not need to consider the case for one or no students.
How would you write this program? Here are some hints:
Step 1: Prompt the user to read the number of the students into variable numberOfStudents.
Step 2: Prompt the user to read first student name into student1 (String).
Step 3: Prompt the user to read first student score into score1 (double).
Step 4: Prompt the user to read second student name into student2 (String).
Step 5: Prompt the user to read second student score into score2 (double).
Step 6: If (score1 < score2), swap student1 with student2 and score1 with score2. Throughout the program, we will use student1 and score1 to store the student name and score with the highest score and student2 and score1 to store the student name and score with the second highest score.
Step 7: Now write a loop to read the next student name and score. Consider the following cases:
Step 7.1: if (score > score1), assign score1 to score2 and score to score1, student1 to student2 and name to student.
Step 7.2: else if (score1 > score2), assign score to score2 and name to student2.
Step 8: Display the top two students' name and score.

Chapter 5: Programming Project 4: Exercise05_23
For this program, you need to use the formula for computing monthlyPayment and totalPayment in Section 2.14.
How would you write this program? Here are some hints:
Step 1: Prompt the user to read loanAmount (double) and numberOfYears (int).
Step 2: Print the header using iomanip, "Interest Rate", "Monthly Payment", "Total Payment";
Step 3: Write a for loop as follows:
Step 3.1: The initial action is to declare and initialize annualInterestRate to 5.0.
Step 3.2: The loop continuation condition is <= 8.0.
Step 3.3: The action after each iteration is annualInterestRate += 1.0/ 8.
Step 3.4: In the loop body, compute monthlyPayment and totalPayment using the formula. Note the monthlyInterestRate is annualInterestRate / 1200. Display annualInterestRate, monthlyPayment, and totalPayment in one line using iomanip to format the output. Please note: in the first column, you need to put three digits after the decimal point. For example, 5.000% is correct, but 5% is wrong.

Chapter 5: Programming Project 5: Exercise05_49
How would you write this program? Here are some hints:
Step 1: Declare and initialize variables numberOfVowels and numberOfConsonants with 0 (int).
Step 2: Prompt the user to enter a string.
Step 3: For each character in the string, do the following:
Step 3.1: Convert the character to uppercase.
Step 3.2: If the character is 'A', 'E', 'I', 'O', or 'U', increase numberOfVowels by 1.
Step 3.3: else if the character is a letter, increase numberOfConsonants by 1.
Step 4: Display the result using the wording as shown in the sample output.

Chapter 5: Programming Project 6: Exercise05_53
How would you write this program? Here are some hints:
Step 1: Prompt the user to enter a string s.
Step 2: Declare a bool variable isValid, initialized it with true.
Step 3: Check if every character in s is a digit or character. If not, assign isValid false.
Step 4: Check if the size of s is less than 8. If so, assign isValid false.
Step 5: Count the number of the digits in s. If the count is less than 2, assign isValid false.
Step 6: Now display `valid password` or `invalid password` based on the value of isValid.

Chapter 6: Programming Project 1: Exercise06_07

How would you write this program? Here are some hints:
The program has two functions: the main function and the futureInvestmentValue function.
Step 1: Implement the futureInvestmentValue function to compute the futureInvestment value from investmentAmount, monthlyInterestRate and years using the formula from Exercise 2.23 in Chapter 2. This function return futureInvestmentValue.
Step 2: Implement the main function as follows:
Step 2.1: Prompt the user to enter investmentAmount (double) and annualInterestRate (double).
Step 2.2: Write a for loop as follows: for each year from 1 to 30, display year and the result from invoking futureInvestmentValue(investmentAmount, annualInterestRate / 1200, year). You need to use iomanip to display formatted output.

Chapter 6: Programming Project 2: Exercise06_13
How would you write this program? Here are some hints:
The program has two functions: the main function and the m(int i) function.
Step 1: Implement the m(int i) function to return the summation as shown in the formula for m(i) given in the description.
Step 1.1: declare and initialize sum.
Step 1.2: for each k from 1 to i, add k / (k + 1.0) to sum. Note we are using 1.0 rather than 1 to obtain a double result.
Step 1.3: return sum.
Step 2: Implement the main function as follows:
Step 2.1: Display the header of the table using iomanip. The header is "i m(i)".
Step 2.2: Write a for loop as follows: for each i from 1 to 20, display i and the result from invoking m(i) using iomanip. The result of m(i) is displayed with four digits after the decimal point. So, 4 should be displayed 4.0000

Chapter 6: Programming Project 3: Exercise06_33
How would you write this program? Here are some hints:
The program has two functions: the main function and the solveEquation function.

```
void solveEquation(double a, double b, double c, double d,
  double e, double f, double& x, double& y, bool& isSolvable)
```

Step 1: Implement the solveEquation function to compute isSolvable, x, and y. Please note that this function should not print anything.
Step 1.1: set isSolvable to false if (a * d – b * c) is 0; true otherwise.
Step 1.2: if isSolvable, compute x and y.
Step 2: Prompt the user to read input a, b, c, d, e, f. Declare bool variable isSolvable, and double variables x and y. Invoke solveEquation. If isSolvable, display x and y; otherwise display "The equation has no solution".

Please note you have to create an array using a constant size. For all the Programming Projects, assume that the maximum number of elements in the array is 100.

Chapter 7: Programming Project 1: Exercise07_01
How would you write this program? Here are some hints:
Assume the maximum number of students is 100.
Step 1: Create an array for scores (double) with the size 100.
Step 2: Declare and initialize variable best to keep the best score. Set the initial value to 0.
Step 3: Prompt the user to enter the scores in a loop that executes numberOfStudents times. For each score entered, store it in scores[i]. Compare it with best. If it is greater than best, assign it to best.
Step 4: Write a for loop for i from 0 to numberOfStudents – 1, compare scores[i] with grade to assign the grade for the student.

Chapter 7: Programming Project 2: Exercise07_03
How would you write this program? Here are some hints:

Step 1: Create an array for counts (int) with size 100. count[0] will store the number of 1s entered and count[99] will store the number of 100 entered. By default, the count[i] is 0.
Step 2: Read a number.
Step 3: Write a while loop:
Step 3.1: The loop continuation condition is (number != 0).
Step 3.2: if number is between 1 and 100, count[number – 1]++.
Step 3.3: read the number again.
Step 4: Write a for loop to display the result as follows:
Step 4.1: for i from 0 to 99
Step 4.2: if counts[i] > 1, display number i + 1, counts[i] and "time" or "times". If (counts[i] > 1), displays "times". If (counts[i] == 1), display "time".

Chapter 7: Programming Project 3: Exercise07_05
How would you write this program? Here are some hints:
Step 1: Create an array for numbers with size 10.
Step 2: Declare and initialize an int variable numberOfDistinctValues.
Step 3: Write a for loop to executed 10 times. Each iteration of the loop performs the following actions:
Step 3.1: Read an int value.
Step 3.2: Test if value is already in numbers.
Step 3.3: if value is not numbers, add value to numbers: numbers[numberOfDistinctValues] = value.
Step 3.4: Increase numberOfDistinctValues by 1.
Step 4: Display the output: display numberOfDistinctValues and all the elements in numbers.

For Step 3.2, you may write a function
bool isInNumbers(int numbers[], int size, int value)
This function return true if value is equal to numbers[0], numbers[1], …, or numbers[size -1].

Chapter 7: Programming Project 4: Exercise07_11
How would you write this program? Here are some hints:
Step 1: Implement the mean(const double x[], int size) function as follows:
Step 2.1: Declare and initialize a double variable sum.
Step 2.2: Write a loop to all elements in array x into sum.
Step 2.3: Return sum / size;
Step 2: Implement the deviation(const double x[], int size) function as follows:
Step 2.1: Declare and initialize squareSum.
Step 2.2: Write a loop. For each element x[i], add (x[i] – mean(x)) ^ 2 to squareSum.
Step 2.3: return sqrt(squareSum / (x.length – 1))
Step 3: Implement the main function as follows:
Step 3.1: Create an array numbers with 10 elements (double).
Step 3.2: Prompt the user to enter 10 numbers and store them in numbers.
Step 3.3: Invoke mean(numbers, 10) and deviation(numbers, 10) to obtain mean and deviation for numbers.

Chapter 7: Programming Project 5: Exercise07_27
Assume that the maximum number of the integers in the list is 100. So declare array of size 100.
How would you write this program? Here are some hints:
Step 1: Implement the isSorted(const int list[], int size) function as follows:
Step 1.1: Write a for loop: for i from 0 to size – 1, if (list[i] > list[i + 1]), return false.
Step 1.2: If nothing is return in the for loop, return true after the for loop.
Step 2: Implement the main function as follows:
Step 2.1: Create list using with **100** elements (int).

Step 2.2: Prompt the user to enter the list size and prompt the user to enter int values for list. Please note that list size is less than or equal to 100. If your input for the program is

```
Enter the size of the list: 8
Enter list: 10 1 5 16 61 9 11 1
```

The size of the list is 8.

Step 3: Invoke isSorted(list, size) to test if the elements in list are sorted.

Chapter 8: Programming Project 1: Exercise08_01
How would you write this program? Here are some hints:
Step 1: Implement the sumColumn(const double m[][SIZE], int rowSize, int columnIndex) function as follows:
Step 1.1: Declare and initialize sum.
Step 1.2: Write a for loop: for (int i = 0; i < rowSize; i++), add m[i][columnIndex] to sum. Note: rowSize is 3.
Step 1.3: Return sum.
Step 2: Implement the main function as follows:
Step 2.1: Declare a two-dimensional array double m[3][4] (double).
Step 2.2: Write a nested for loop to obtain the input into m.
Step 2.3: Write a for loop: for (int j = 0; j < SIZE; j--) and invoke sumColumn(m, j) and display it. Note: column size is 4.
Hint: Make sure you use the correct row and column size. The matrix has 3 rows and 4 columns. If row/column size is incorrect, your output is unpredictable. Your code may pass CheckExerciseTool even though it is incorrect due to the unpredictable nature. Please see CheckPoint See CheckPoint 8.3.1 and 7.2.12 for examples.

Chapter 8: Programming Project 2: Exercise08_05
How would you write this program? Here are some hints:
Step 1: Implement the addMatrix(const double a[][N], const double m2[][N], double c[][N]) function as follows:
Step 1.1: Write a nested for loop to assign a[i][j] + b[i][j] to c[i][j].
Step 2: Implement the printResult(const double m1[][N], const double m2[][N], const double m3[][N], char op) function as follows:
Step 2.1: For each row i from m1.length, display a row in m1, m2, and m3. In the row middle, display the op between m1 and m2 and display the = symbol between m2 and m3.
Step 3: Implement the main function as follows: (numbers are double)
Step 3.1: Create array m1. Enter input for m1.
Step 3.2: Create array m2. Enter input for m2.
Step 3.3: Create array m3. Invoke addMatrix(m1, m2, m3).
Step 3.4: Display the result by invoking printResult(m1, m2, m3)

Chapter 8: Programming Project 3: Exercise08_11

The spaces that separate the 3 by 3 matrices in the project description are missing. Here are the correct examples:

Here are some examples:

```
0 0 0    1 0 1    1 1 0    1 0 1    1 0 0

0 1 0    0 0 1    1 0 0    1 1 0    1 1 1

0 0 0    1 0 0    0 0 1    1 0 0    1 1 0
```

How would you write this program? Here are some hints:

Step 1: Implement dec2Bianry(short decimal) to return a binary string of size 9. For example, decimalToBinaryary(15) returns "000001111". The function header is
    string dec2Bianry(short decimal)

Step 2: Implement string2Array(const string& s, char result[][3]) to obtain a 3 by 3 array result of characters consisting of H and T. For example, after invoking string2Array("000001111", result), result is {{'H', 'H', 'H'}, {'H', 'H', 'T'}, {'T', 'T', 'T'}}. The function header is

void string2Array(const string& s, char result[][3])

Step 3: Implement the main function.
Step 3.1: Prompt the user to enter a decimal short int.
Step 3.2: Invoke decimalToBinaryary to return a binary string.
Step 3.3: Invoke string2Array to obtain a 3 by 3 array of characters.
Step 3.4: Display the 3 by 3 array.

Chapter 8: Programming Project 4: Exercise08_23
How would you write this program? Here are some hints:
Step 1: Implement the main function.
Step 1.1: Set MAX_NUMBER_OF_CITIES to 20. Create cities using double cities[MAX_NUMBER_OF_CITIES][2].
Step 1.2: Prompt the user to enter numberOfCities.
Step 1.3: Prompt the user to enter the coordinates for the cities.
Step 1.4: Declare minTotal (double) and minIndex (int) to store the minimum total distance and the index of the minimum total distance city.
Step 1.5: For every city with index i, invoke totalDistance(cities, i) to return the totalDistance. If it is < minTotal, assign totalDistance to minTotal and i to minIndex.
Step 1.6: Display the minTotal and minIndex for the central city.
Step 2: Implement getDistance(const double c1[], const double c2[]). This function returns the distance between (c1[0], c1[1]) and (c2[0], c2[1]).
Step 3: Implement totalDistance(const double cities[][2], int numberOfCities, int i). This function returns the total distance from city i to all other cities.

Chapter 9: Programming Project 1: Exercise09_01
Please note that you have to put all code in one file in order to submit to REVEL. To combine the .h, .cpp, and the main function in one file, see FAQ Q2.

Chapter 9: Programming Project 1: Exercise09_11
Please note that you have to put all code in one file in order to submit to REVEL. To combine the .h, .cpp, and the main function in one file, see FAQ Q2.

Chapter 10: Programming Project 1: Exercise10_01
How would you write this program? Here are some hints:
Step 1: Implement the main() function as follows:
Step 1.1: Read string s1.
Step 1.2: Read string s2.
Step 1.3: Invoke isAnagram(s1, s2) to test if s1 and s2 are anagrams.
Step 2: Implement the isAnagram(s1, s2) function as follows:
Step 2.1: If s1 and s2 have different sizes, return false.
Step 2.2: Sort s1 and s2 and return true if they are the same.
Step 3: Implement the sort(string& s) function to sort a string as follows:

Chapter 10: Programming Project 1: Exercise10_13
Please note that you have to put all code in one file in order to submit to REVEL. To combine the .h, .cpp, and the main function in one file, see FAQ Q2.

Chapter 11: Programming Project 1: Exercise11_11
Use the template in https://liangcpp.pearsoncmg.com/test/Exercise11_11.txt to write your code.

Chapter 12: Programming Project 1: Exercise12_13
Use the template in https://liangcpp.pearsoncmg.com/test/Exercise12_13.txt to write your code.

Study the code in Listing 12.12 EvaluateExpression.cpp using some samples, for example, 1+2, 2 * 3 - 3, etc.

Modify the example EvaluateExpression.cpp incrementally. Once step at a time and you will know which step you are struggling.
Step 1. The operator % can be implemented similar to the * and / operators. Add the code for processing % in lines 92-103 in Listing 12.12.
Step 2. The operator ^ has the highest precedence. However, note that the ^ operator is right-associative, meaning that 2 ^ 3 ^ 2 is same as 2 ^ (3 ^ 2). In lines 92-103 in Listing 12.12, the program processes the * and / operators, add the code for processing the ^ operator after this block.

Listing 12.12 EvaluateExpression.cpp

Modify the code in the text. 1. Read the input expression from the console rather than passing from the command-line. 2. The operator % can be implemented similar to the * and / operators. 3. The operator ^ has the highest precedence. However, note that the ^ operator is right-associative, meaning that 2 ^ 3 ^ 2 is same as 2 ^ (3 ^ 2).

Chapter 13: Programming Project 1: Exercise13_03
Due to my server security restriction, the user is not allowed to read/write files on the server. You cannot test Exercise13_03 from CheckExercise. However, you can test your code using the following two cases.

Case 1. Suppose the file Exercise13_3.txt contains numbers 3 9\n 8. The sample run of the program is shown as follows:

<sample run>
Total is 20
Average is 6.66667
<end sample run>

Case 2. Suppose the file Exercise13_3.txt contains numbers 1\n 2 3. The sample run of the program is shown as follows:

<sample run>
Total is 6
Average is 2.0
<end sample run>

Chapter 13: Programming Project 2: Exercise13_01
Due to my server security restriction, the user is not allowed to read/write files on the server. You cannot test this program from CheckExercise. However, you can verify it before submitting the REVEL. After your program is executed, check Exercise13_1.txt. The file should contain 0 1 2 3 4 5 6 ... 93 94 95 96 97 98 99 at the end.

Chapter 14: Programming Project 1: Exercise14_07
Use the template in https://liangcpp.pearsoncmg.com/test/Exercise14_07.txt to write your code. When you submit it to REVEL, only select the code that is enclosed between the BEING SUBMISSION comment line and the END SUBMISSION comment line in the template.

Chapter 15: Programming Project 1: Exercise15_03
Use the template in https://liangcpp.pearsoncmg.com/test/Exercise15_03.txt to write your code.

Chapter 16: Programming Project 1: Exercise16_01
Use the template in https://liangcpp.pearsoncmg.com/test/Exercise16_01.txt to write your code.

Chapter 16: Programming Project 1: Exercise16_05

Use the template in https://liangcpp.pearsoncmg.com/test/Exercise16_05.txt to write your code.

Chapter 17: Programming Project 3: Exercise17_11
Hint:
The program contains the main function and the sumDigits(long n). The main function prompts the user for the input number n. It then invokes sumDigits(n) to return the sum of the digits in n.

sumDigits(n) returns n if n is a single digit. Otherwise, it returns sumDigits(n / 10) + sumDigits(n % 10). n % 10 is the last digit in n. n / 10 is the remaining number after removing the last digit.

Chapter 17: Programming Project 4: Exercise17_15
Hint:
The program contains the main function and two overloaded count functions. The main function prompts the user for the input string and then a character. It then invokes the count(const string& s, char a) function.

The count(const string& s, char ch) function invokes count(s, ch, s.length() - 1).

The count(s, ch, high) is a recursive helper function. The function returns 0 if high < 0. Otherwise, it returns count(s, ch, high - 1) + (s[high] == ch ? 1 : 0).

Chapter 17: Programming Project 5: Exercise17_21
Hint:
For simplicity, you can assume that the decimal integer is greater than 0. Your submission will work with this assumption.

The program contains the main function and the decimalToBinary function. The main function prompts the user to enter an integer then invokes the decimalToBinary(int) to return a binary string for the integer. It displays the binary string.

The decimalToBinary(int value) function returns "" if value is 0, otherwise, it returns decimalToBinary(value / 2) + (value % 2 == 0? "0" : "1").

Note that decimalrBinary(value / 2) returns a string. value % 2 is 0 or 1.

Chapter 18: Programming Project 3: Exercise18_05

Correction:
The Sample Run should be

Sample Run

```
Enter a series of numbers ending with 0: 2 4 4 8 8 8 8 2 4 4 0

The longest same number sequence starts at index 3 with 4 values of 8
```