APPENDIX E Answers to Checkpoint Questions

Chapter 1

1.1 A program is a set of instructions that a computer follows to perform a task.
1.2 Hardware is all of the physical devices, or components, that a computer is made of.
1.3 The central processing unit (CPU), main memory, secondary storage devices, input devices, and output devices.
1.4 The CPU
1.5 Main memory
1.6 Secondary storage
1.7 Input device
1.8 Output device
1.9 One byte
1.10 A bit
1.11 The binary numbering system.
1.12 It is an encoding scheme that uses a set of 128 numeric codes to represent the English letters, various punctuation marks, and other characters. These numeric codes are used to store characters in a computer's memory. (ASCII stands for the American Standard Code for Information Interchange.)
1.13 Unicode
1.14 Digital data is data that is stored in binary, and a digital device is any device that works with binary data.
1.15 Machine language
1.16 Main memory, or RAM
1.17 The fetch-decode-execute cycle.
1.18 It is an alternative to machine language. Instead of using binary numbers for instructions, assembly language uses short words that are known as mnemonics.
1.19 A high-level language
1.20 Syntax
1.21 A compiler
1.22 An interpreter
1.23 A syntax error
1.24 The operating system
1.25 A utility program
1.26 Application software

Chapter 2

2.1 Any person, group, or organization that is asking you to write a program.
2.2 A single function that the program must perform in order to satisfy the customer.
2.3 A set of well-defined logical steps that must be taken to perform a task.
2.4 An informal language that has no syntax rules, and is not meant to be compiled or executed. Instead, programmers use pseudocode to create models, or "mock-ups" of programs.
2.5 A diagram that graphically depicts the steps that take place in a program.
2.6 Ovals are terminal symbols. Parallelograms are either output or input symbols. Rectangles are processing symbols.
2.7 Input, processing, and output.
2.8 A chart, or table, that describes a program's input, processing, and output.
2.9 A set of statements that execute in the order that they appear.
2.10 A string is a sequence of characters that is used as data. A string literal is a string that appears in the actual code of a program.
2.11 Quotation marks
2.12 A storage location in memory that is represented by a name.
2.13

- Variable names must be one word. They cannot contain spaces.
- In most languages, punctuation characters cannot be used in variable names. It is usually a good idea to use only alphabetic letters and numbers in variable names.
- In most languages, the first character of a variable name cannot be a number.

2.14 camelCase
2.15

- The program pauses and waits for the user to type something on the keyboard, and then press the Enter key.
- When the Enter key is pressed, the data that was typed is stored in the `temperature` variable.

2.16 Any hypothetical person that is using a program and providing input for it.
2.17 A message that tells (or asks) the user to enter a specific value.
2.18

1. Display a prompt on the screen.
2. Read a value from the keyboard.

2.19 The term *user-friendly* is commonly used in the software business to describe programs that are easy to use.
2.20 A statement that sets a variable to a specified value.
2.21 It is replaced.
2.22

1. Perform any operations that are enclosed in parentheses.
2. Perform any operations that use the exponent operator to raise a number to a power.
3. Perform any multiplications, divisions, or modulus operations as they appear from left to right.
4. Perform any additions or subtractions as they appear from left to right.

2.23 It raises a number to a power.
2.24 It performs division, but instead of returning the quotient it returns the remainder.

2.25 The variable's name and data type.
2.26 Yes, you must write a variable declaration before any other statement that uses the variable.
2.27 The assignment of a value to a variable at the time the variable is declared.
2.28 Yes, they are a common cause of errors. If an uninitialized variable is used in an operation such as a calculation, a logic error will occur.
2.29 A variable that has been declared, but has not been initialized or assigned a value.
2.30 External documentation is typically designed for the user. It consists of documents such as a reference guide that describes the program's features, and tutorials that teach the user how to operate the program.
2.31 Internal documentation appears as comments in a program's code. Comments are short notes placed in different parts of a program, explaining how those parts of the program work.
2.32 Programmers generally write block comments and line comments. Block comments take up several lines and are used when lengthy explanations are required. Line comments are comments that occupy a single line, and explain a short section of the program.

Chapter 3

3.1 A module is a group of statements that exist within a program for the purpose of performing a specific task.
3.2 A large task is divided into several smaller tasks that are easily performed.
3.3 If a specific operation is performed in several places in a program, a module can be written once to perform that operation, and then be executed any time it is needed.
3.4 Modules can be written for the common tasks that are needed by the different programs. Those modules can then be incorporated into each program that needs them.
3.5 When a program is developed as a set of modules that each perform an individual task, then different programmers can be assigned the job of writing different modules.
3.6 In most languages, a module definition has two parts: a header and a body. The header indicates the starting point of the module, and the body is a list of statements that belong to the module.
3.7 To call a module means to execute the module.
3.8 When the end of the module is reached, the computer jumps back to the part of the program that called the module, and the program resumes execution at that point.
3.9

- The overall task that the program is to perform is broken down into a series of subtasks.
- Each of the subtasks is examined to determine whether it can be further broken down into more subtasks. This step is repeated until no more subtasks can be identified.
- Once all of the subtasks have been identified, they are written in code.

3.10 A local variable is a variable that is declared inside a module. It belongs to the module in which it is declared, and only statements inside the same module can access it.
3.11 The part of a program in which a variable may be accessed.
3.12 No, it is not permissible. The compiler or interpreter would not know which variable to use when a statement tries to access one of them.
3.13 Yes, it is permissible.
3.14 Arguments
3.15 Parameters
3.16 Yes, an error will usually occur if an argument's data type is different from the data type of the parameter it is being passed to.
3.17 A parameter variable's scope is usually the entire module in which the parameter is declared.
3.18 Passing an argument by value means that only a copy of the argument's value is passed into the parameter variable. If the contents of the parameter variable are changed inside the module, it has no effect on the argument in the calling part of the program. Passing an argument by reference means that the argument is passed into a special type of parameter known as a reference variable. When a reference variable is used as a parameter in a module, it allows the module to modify the argument in the calling part of the program.
3.19 The entire program
3.20 Here are three:

- Global variables make debugging difficult. Any statement in a program can change the value of a global variable. If you find that the wrong value is being stored in a global variable, you have to track down every statement that accesses it to determine where the bad value is coming from. In a program with thousands of lines of code, this can be difficult.
- Modules that use global variables are usually dependent on those variables. If you want to use such a module in a different program, you will most likely have to redesign it so it does not rely on the global variable.

Global variables make a program hard to understand. A global variable can be modified by any statement in the program. If you are to understand any part of the program that uses a global variable, you have to be aware of all the other parts of the program that access the global variable.

3.21 A global constant is a named constant that is available to every module in the program. It is permissible to use global constants. Because a global constant's value cannot be changed during the program's execution, you do not have to worry about its value being altered.

Chapter 4

4.1 A logical design that controls the order in which a set of statements executes.
4.2 It is a program structure that can execute a set of statements only under certain circumstances.
4.3 A decision structure that provides a single alternative path of execution. If the condition that is being tested is true, the program takes the alternative path.
4.4 An expression that can be evaluated as either true or false.
4.5 You can determine whether one value is greater than, less than, greater than or equal to, less than or equal to, equal to, or not equal to another value.
4.6

```
If y == 20 Then
   Set x = 0
End If
```

4.7

```
If sales >= 10000 Then
   Set commission = 0.2
End If
```

4.8 A dual alternative decision structure has two possible paths of execution—one path is taken if a condition is true, and the other path is taken if the condition is false.
4.9 ***`If-Then-Else`***
4.10 When the condition is false.
4.11
z is not less than a.
4.12

```
Boston
New York
```

4.13 A dual alternative decision structure has two possible paths of execution—one path is taken if a condition is true, and the other path is taken if the condition is false.
4.14
If-Then-Else
4.15 If the condition is false.
4.16

```
If number == 1 Then
   Display "One"
Else If number == 2 Then
   Display "Two"
Else If number == 3 Then
   Display "Three"
Else
   Display "Unknown"
End If
```

4.17 A structure that tests the value of a variable or an expression and then uses that value to determine which statement or set of statements to execute.
4.18 With a `Select Case` statement.
4.19 A variable or an expression.
4.20 In such an event, you can use the `If-Then-Else If` statement, or a nested decision structure.
4.21 It is an expression that is created by using a logical operator to combine two Boolean subexpressions.
4.22
F

T

F

F

T

T

T

F

F

T

4.23
T

F

T

T

T

4.24 The `AND` operator: If the expression on the left side of the `AND` operator is false, the expression on the right side will not be checked.
The `OR` operator: If the expression on the left side of the `OR` operator is true, the expression on the right side will not be checked.
4.25

```
 If speed >= 0 AND speed <= 200 Then
    Display "The number is valid"
End If
```

4.26

```
 If speed < 0 OR speed > 200 Then
    Display "The number is not valid"
End If
```

4.27 True or false
4.28 A variable that signals when some condition exists in the program.

Chapter 5

5.1 A structure that causes a section of code to repeat.
5.2 A loop that uses a true/false condition to control the number of times that it repeats.
5.3 A loop that repeats a specific number of times.
5.4 An execution of the statements in the body of the loop.
5.5 A pretest loop tests its condition before it performs an iteration. A posttest loop tests its condition after it performs an iteration.
5.6 Before
5.7 After
5.8 A loop that has no way of stopping, and repeats until the program is interrupted.
5.9 A `Do-While` loop iterates while a condition is true. When the condition is false, the `Do-While` loop stops. A `Do-Until` loop iterates until a condition is true. When the condition is true, the `Do-Until` loop stops.
5.10 A variable that is used to store the number of iterations that it has performed.
5.11 Initialization, test, and increment.
5.12 Incrementing a variable means increasing its value. Decrementing a variable means decreasing its value.
5.13 6
5.14

```
1

2

3

4

5
```

5.15

```
0

100

200

300

400

500
```

5.16

```
1

2

3

4

5

6

7
```

8

5.17
1

3

5

7

5.18
5

4

3

2

1

5.19

1. A loop that reads each number in the series.
2. A variable that accumulates the total of the numbers as they are read.

5.20 A variable that is used to accumulate the total of a series of numbers.
5.21 Yes, it should be initialized with the value 0. This is because values are added to the accumulator by a loop. If the accumulator does not start at the value 0, it will not contain the correct total of the numbers that were added to it when the loop ends.
5.22 15
5.23 5
5.24 A sentinel is a special value that marks the end of a list of values.
5.25 A sentinel value must be unique enough that it will not be mistaken as a regular value in the list.

Chapter 6

6.1 When a module finishes, the program merely returns back to the part of the program that called the module, and execution resumes at that point. When a function finishes, it returns a value back to the part of the program that called it.

6.2 A prewritten function that comes with a programming language.

6.3 The term *black box* is used to describe any mechanism that accepts input, performs some operation that cannot be seen on the input, and produces output. A library function can be regarded as a black box because you cannot see the code inside the function. The function accepts input, performs an operation on the input, and produces output.

6.4 It assigns a random number in the range of 1 through 100 to the `x` variable.

6.5 It displays a random number in the range of 1 through 20.

6.6 The `Return` statement specifies the value that the function returns to the part of the program that called the function. When the `Return` statement is executed, it causes the function to terminate and return the specified value.

6.7

a. `doSomething`
b. `Integer`
c. 10

6.8 A function that returns either true or false.

Chapter 7

7.1 It means that if bad data (garbage) is provided as input to a program, the program will produce bad data (garbage) as output.
7.2 When input is given to a program, it should be inspected before it is processed. If the input is invalid, then it should be discarded and the user should be prompted to enter the correct data.
7.3 The input is read, and then a pretest loop is executed. If the input data is invalid, the body of the loop executes. In the body of the loop, an error message is displayed so the user will know that the input was invalid, and then the new input is read. The loop repeats as long as the input is invalid.
7.4 It is the input operation that takes place just before an input validation loop. The purpose of the priming read is to get the first input value.
7.5 None

Chapter 8

8.1 No, you cannot. All of the items in an array must be of the same data type.
8.2 A nonnegative integer that specifies the size of an array.
8.3 No
8.4 An individual storage location in an array.
8.5 A number that identifies a specific element in an array.
8.6 0
8.7

a. `numbers`
b. 7
c. `Real`
d. 6

8.8 Many languages support array bounds checking, which means they do not allow a program to use an invalid array subscript.
8.9 An off-by-one error occurs when a loop iterates one time too many or one time too few.
8.10 An algorithm developed for the purpose of locating a specific item in a larger collection of data, such as an array.
8.11 The first element in the array.
8.12 The loop sequentially steps through each element in the array, comparing the elements to the value being searched for. When the value is found, the loop stops.
8.13 It looks at every element in the array.
8.14 You use a function similar to the `contains` function described in this chapter. The `contains` function returns true if a string is found inside another string, or false otherwise.
8.15 To calculate the total of the values in an array, you use a loop with an accumulator variable. The loop steps through the array, adding the value of each array element to the accumulator.
8.16 The first step in calculating the average of the values in an array is to get the sum of the values. You use the algorithm for totaling the values in an array to perform this. The second step is to divide the sum by the number of elements in the array.
8.17 You create a variable to hold the highest value. In the examples shown in this book, the variable is named `highest`. Then, you assign the value at element 0 to the `highest` variable. Next, you use a loop to step through the rest of the array elements, beginning at element 1. Each time the loop iterates, it compares an array element to the `highest` variable. If the array element is greater than the `highest` variable, then the value in the array element is assigned to the `highest` variable. When the loop finishes, the `highest` variable will contain the highest value in the array.
8.18 You create a variable to hold the lowest value. In the examples shown in this book, the variable is named `lowest`. Then, you assign the value at element 0 to the `lowest` variable. Next, you use a loop to step through the rest of the array elements, beginning at element 1. Each time the loop iterates, it compares an array element to the `lowest` variable. If the array element is less than the `lowest` variable, then the value in the array element is assigned to the `lowest` variable. When the loop finishes, the `lowest` variable will contain the lowest value in the array.
8.19 You assign the individual elements of the array that you are copying to the elements of the other array. This is usually best done with a loop.
8.20 You use the same subscript to access data items in the two arrays.

8.21 It would be stored in `creditScore[82]`

8.22 88 rows and 100 columns

8.23 `Set points[87][99] = 100`

8.24

```
Constant Integer ROWS = 3
Constant Integer COLS = 5
Declare Integer table[ROWS][COLS] = 12, 24, 32, 21, 42,
                                    14, 67, 87, 65, 90,
                                    19,  1, 24, 12,  8
```

8.25

```
Declare Integer row
Declare Integer col
For row = 0 To ROWS - 1
  For col = 0 to COLS - 1
     Set info[row][col] = 99
  End For
End For
```

8.26

```
Constant Integer RACKS = 50
Constant Integer SHELVES = 10
Constant Integer BOOKS = 25
Declare String books[RACKS][SHELVES][BOOKS]
```

Chapter 9

9.1 The bubble sort
9.2 The insertion sort algorithm
9.3 The selection sort algorithm
9.4 The sequential search algorithm simply uses a loop to step through each element of an array, comparing each element's value with the value being searched for. The binary search algorithm, which requires the values in the array to be sorted in order, starts searching at the element in the middle of the array. If the middle element's value is greater than the value being searched for, the algorithm next tests the element in the middle of the first half of the array. If the middle element's value is less than the value being searched for, the algorithm next tests the element in the middle of the last half of the array. Each time the array tests an array element and does not find the value being searched for, it eliminates half of the remaining portion of the array. This method continues until the value is found, or there are no more elements to test. The binary search is more efficient than the sequential search.
9.5 500
9.6 10

Chapter 10

10.1 On the computer's disk.
10.2 A file that a program writes data to. It is called an output file because the program sends output to it.
10.3 A file that a program reads data from. It is called an input file because the program receives input from it.
10.4

1. Open the file.
2. Process the file.
3. Close the file.

10.5 Text and binary. A text file contains data that has been encoded as text, using a scheme such as Unicode. Even if the file contains numbers, those numbers are stored in the file as a series of characters. As a result, the file may be opened and viewed in a text editor such as Notepad. A binary file contains data that has not been converted to text. As a consequence, you cannot view the contents of a binary file with a text editor.
10.6 Sequential and direct access. When you work with a sequential access file, you access data from the beginning of the file to the end of the file. When you work with a direct access file, you can jump directly to any piece of data in the file without reading the data that comes before it.
10.7 The file's external name and internal name. The external name is the file name that identifies the file on the disk. The internal name is like a variable name. It identifies the file in your program code.
10.8 The file's contents are erased.
10.9 Opening a file creates a connection between the file and the program. It also creates an association between the file and its internal name.
10.10 Closing a file disconnects the file from the program.
10.11 A predefined character or set of characters that marks the end of piece of data. In many languages, a delimiter is written after each item that is stored in a file.
10.12 A special character, or set of characters, known as the end-of-file marker.
10.13 A file's read position marks the location of the next item that will be read from the file. When an input file is opened, its read position is initially set to the first item in the file.
10.14 You open the file in append mode. When you write data to a file in append mode, the data is written at the end of the file's existing contents.
10.15

```
Declare Integer counter
Declare OutputFile myFile
Open myFile "myfile.dat"
For counter = 1 To 10
   Write myFile, counter
End For
Close myFile
```

10.16 The `eof` function determines whether the end of a file has been reached.
10.17 No, this usually causes an error.
10.18 It would mean that the program had reached the end of the file associated with the name `myFile`.

10.19 b
10.20 A record is a complete set of data that describes one item, and a field is a single piece of data within a record.
10.21 You copy all of the original file’s records to the temporary file, but when you get to the record that is to be modified, you do not write its old contents to the temporary file. Instead, you write its new, modified values to the temporary file. Then, you finish copying any remaining records from the original file to the temporary file.
10.22 You copy all of the original file’s records to the temporary file, except for the record that is to be deleted. The temporary file then takes the place of the original file. You delete the original file and rename the temporary file, giving it the name that the original file had on the computer’s disk.

Chapter 11

11.1 A menu-driven program displays a list of operations that it can perform on the screen, and allows the user to select the operation that he or she wants the program to perform. The list of operations that is displayed on the screen is called a *menu*.
11.2 The user types the character that corresponds to the menu item that he or she wants to select.
11.3 You use a decision structure of some type. You might choose a case structure, nested `If-Then-Else` statements, or an `If-Then-Else If` statement.
11.4 Without a loop, the program would end after the selected action is performed. This would be an inconvenience because it would require the user to rerun the program to perform another action.
11.5 Ending the program should be an item that the user can select from the menu. When the user selects this item, the loop stops and the program ends.
11.6 In a program that uses a single-level menu, all of the menu selections fit nicely in a single menu. When the user selects an operation from the menu, the program immediately performs that operation and then the program redisplays the menu.
11.7 A program that uses a multiple-level menu typically displays a *main menu* when the program starts, showing only a few items, and then displays smaller *submenus* when the user makes a selection.
11.8 Users often have trouble sorting through the items in a menu when given too many choices.

Chapter 12

12.1 y

12.2 ***Set str[0] =“L”***

12.3

```
If isDigit(str[0]) Then
   delete(str, 0, 0)
End If
```

12.4

```
If isUpperCase(str[0]) Then
   Set str[0] = "0"
End If
```

12.5 ***insert(str, 0,"Hello ")***

12.6 ***delete(city, 0, 2)***

Chapter 13

13.1 A recursive algorithm requires multiple method calls. Each method call requires several actions to be performed by the computer. These actions include allocating memory for parameters and local variables, and storing the address of the program location where control returns after the method terminates. All of these actions are known as overhead. In an iterative algorithm, which uses a loop, such overhead is unnecessary.
13.2 A case in which the problem can be solved without recursion.
13.3 A case in which the problem is solved using recursion.
13.4 When it reaches the base case.
13.5 In direct recursion, a recursive method calls itself. In indirect recursion, method A calls method B, which in turn calls method A.

Chapter 14

14.1 An object is a software entity that contains both data and procedures.
14.2 Encapsulation is the combining of data and code into a single object.
14.3 When an object's internal data is hidden from outside code and access to that data is restricted to the object's methods, the data is protected from accidental corruption. In addition, the programming code outside the object does not need to know about the format or internal structure of the object's data.
14.4 Public methods can be accessed by entities outside the object. Private methods cannot be accessed by entities outside the object. They are designed to be accessed internally.
14.5 The metaphor of a blueprint represents a class.
14.6 Objects are the cookies.
14.7 A key word that specifies how code outside a class can access a field or method.
14.8 `Private`
14.9 The memory address of the object that it references.
14.10 It creates an object in the computer's memory.
14.11 An accessor is a method that gets a value from a class's field but does not change it. A mutator is a method that stores a value in a field or changes the value of a field in some other way.
14.12 A constructor is a method that typically initializes an object's fields. A constructor executes when an object is created.
14.13 If you do not write a constructor in a class, most languages automatically provide one when the class is compiled. The constructor that is automatically provided is usually known as the default constructor. Typically, the default constructor assigns default starting values to the object's fields.
14.14 The top section is where you write the name of the class. The middle section holds a list of the class's fields. The bottom section holds a list of the class's methods.
14.15 By writing a colon followed by `String` **after the name of the field. Here is an example:** `description : String`
14.16 You use a minus sign to indicate private specification, and a plus sign to indicate public specification.
14.17 A written description of the real-world objects, parties, and major events related to the problem.
14.18 First, identify the nouns, pronouns, and noun phrases in the problem domain description. Then, refine the list to eliminate duplicates, items that you do not need to be concerned with in order to solve the problem, items that represent objects instead of classes, and items that represent simple values that can be stored in variables.
14.19 The things that the class is responsible for knowing, and the actions that the class is responsible for doing.
14.20 When the value of an item is dependent on other data and that item is not updated when the other data is changed, it is said that the item has become stale.
14.21 The superclass is the general class and the subclass is the specialized class.
14.22 When an "is a" relationship exists between objects, it means that the specialized object has all of the characteristics of the general object, plus additional characteristics that make it special.
14.23 The superclass's fields and methods, except those that are private.
14.24 `Bird` **is the superclass and** `Canary` **is the subclass.**
14.25

```
I'm a potato.
I'm a potato.
```

Chapter 15

15.1 The part of a computer and its operating system that the user interacts with.
15.2 A command line interface typically displays a prompt, and the user types a command, which is then executed.
15.3 The program.
15.4 A program that responds to events that take place, such as the user clicking a button.
15.5 A diagram that shows how the program flows from one window to the next as the user interacts with it.
15.6 Because programmers had to write code that constructed the windows, create graphical elements such as icons and buttons, and set each element's color, position, size, and other properties. Even a simple GUI program that displayed a message such as "Hello world" required the programmer to write a hundred or more lines of code. Furthermore, the programmer could not actually see the program's user interface until the program was compiled and executed.
15.7 You drag the item from a "toolbox" to the window editor.
15.8 An item that appears in a program's graphical user interface.
15.9 A component's name identifies the component in the program, in the same way that a variable's name identifies the variable.
15.10 A component's properties determine how the component appears on the screen.
15.11 An action that takes place within a program, such as the clicking of a button.
15.12 A module that automatically executes when a specific event occurs.
15.13

a. It responds to a *`Click`* event.
b. The component's name is *`showValuesButton`*.

15.14 It executes automatically when a GUI application (or mobile app) starts running
15.15 Less
15.16 Larger
15.17 Here are five unique capabilities that mobile devices usually have, compared to desktop computers:

- make and receive phone calls
- send and receive text messages
- determine the device's location
- detect the device's physical orientation in 3D space
- determine when the device is moving

15.18 Non-visual components are objects that you cannot see in a GUI, but provide special capabilities to your programs. Examples that we discussed are the PhoneCall component, the TextMessage component, and the Location component.
15.19 An *`IncomingCall`* event is generated.
15.20 An *`IncomingText`* event is generated.
15.21 A *`LocationChanged`* event is generated.