Answers to Review Questions

Chapter 3

Fill-in-the-Blank

1. TextBox
2. `txt`
3. String concatenation
4. Line-continuation
5. Line-continuation
6. Focus
7. Tab order
8. TabIndex
9. TabStop
10. Alt
11. Text
12. Variable
13. Variable declaration
14. Data type
15. Local
16. Type conversion or type mismatch
17. Function
18. `CInt`
19. `"c"`
20. Argument
21. Precedence
22. Named constant

23. GroupBox control

24. Load

25. Breakpoint

True or False

1. True

2. True

3. False

4. True

5. False

6. True

7. False

8. True

9. False

10. True

11. True

12. False

13. False

14. False

15. True

16. False

17. True

18. True

19. False

20. True

21. True

22. True

23. False

24. False

Multiple Choice

1. b

2. a

3. d

4. b

5. c

6. a

7. d

8. c

9. b

10. c

11. b

12. c

Short Answer

1. The Label control's Text property is for displaying information only--the user cannot directly alter its contents. The TextBox control's Text property, however, is for input purposes. The user can alter it by typing characters into the text box control. Whatever the user types into the text box is stored in its Text property.

2. By calling the text box's `Clear` method, such as:

```
txtInput.Clear()
```

   You can also clear a text box by storing an empty string in its Text property. You do this with a statement similar to:

```
txtInput.Text = ""
```

3. When an application is running and a form is displayed, one of the form's controls always has the focus. The control that has the focus is the one that receives the user's keyboard input or mouse clicks.

4. `txtPassword.Focus()`

5. When you create a control, Visual Basic automatically assigns a value to its TabIndex property. The first control you create on a form will have a TabIndex of 0. The second control will have a TabIndex of 1, and so forth. The control with a TabIndex of 0 will be the first control in the tab order. The next control in the tab order will be the one who's TabIndex is 1. The tab order continues in this sequence.

6. The TabIndex property contains a numeric value, which indicates the control's position in the tab order. The control with a TabIndex of 0 will be the first control in the tab order. The next control in the tab order will be the one who's TabIndex is 1. The tab order continues in this sequence.

7. You assign an access key to a button through its Text property. For example, assume an application has a button named `btnExit`. You wish to assign the access key *Alt+X* to the button, so the user may trigger the button's `Click` event by pressing *Alt+X* on the keyboard. To make the assignment you must place an ampersand (&) before the letter x in the button's Text property. The Text property will contain the value: E&xit.

8. It causes the letter that immediately follows the ampersand (in the Text property) to appear underlined.

9. The Single data type stores floating point numbers, while the Integer data type stores whole numbers.

10. The following are recommended variable names for each of the information items. Although your names may not be the same, they should give an indication of the variable's purpose.

    a. `intBackPacksSold`
    b. `dblPoundsDogFood`
    c. `dtmToday`
    d. `decWholesale`
    e. `strCustomerName`
    f. `dblGalaxyDistance`
    g. `intMonthNumber`

11. A string variable's default value is `Nothing`. The value `Nothing` is invalid for many operations and can cause a run-time error.

12. When the procedure that it was declared in ends.

13. Strings converted by the `CDec` function:

    a. The string `"22.9000"` will be converted to the Decimal value 22.9000
    b. The string `"1xfc47uvy"` is invalid.
    c. The string `"$19.99"` will be converted to the Decimal value 19.99
    d. The string `"0.05%"` is invalid
    e. `String.Empty` is invalid

14. The `CDec` function attempts to convert the string to a Decimal value. If an invalid character is found in the string that prevents the conversion from taking place, the function throws an exception.

15.

Expression	Value
`5 + 2 * 8`	21
`20 / 5 - 2`	2
`4 + 10 * 3 - 2`	32
`(4 + 10) * 3 - 2`	40

16.

Function Call	Return Value
`dblTest.ToString("d2")`	An exception was thrown
`dblTest.ToString("c2")`	"$67,521.58"
`dblTest.ToString("e1")`	"6.8e+004"
`dblTest.ToString("f2")`	"67521.58"

17. You select multiple controls by using one of the following techniques:
    - Position the mouse cursor over an empty part of the form that is near the controls you wish to select. Click and drag a selection box around the controls.
    - When you release the mouse button, all the controls that were partially or completely enclosed in the selection box will be selected.
    - Hold the *Ctrl* key down while clicking each control you wish to select.

18. The question only asks for three ways, but here are four possible ways to set breakpoints:
    - Place the mouse cursor in the left margin of the *Code* window next to the line of code you wish to set as the breakpoint and click the left mouse button
    - Move the cursor to the line you wish to set as a breakpoint and press the *F9* key
    - Move the cursor to the line you wish to set as a breakpoint, select *DEBUG* from the menu bar, and then select *Toggle Breakpoint* from the menu
    - If the *Debug* tool bar is visible, move the cursor to the line you wish to set as a breakpoint, and then click the *Toggle Breakpoint* button.

What Do You Think?

1. For flexibility - In some instances, the programmer may not want spaces inserted between two joined strings.

2. Code is easier to read when you do not have to scroll horizontally to see an entire line.

3. Because they do not receive input.

4. Disorganized tab orders can confuse and irritate the users of your application. Users want to tab smoothly from one control to the next, in a logical sequence.

5. Some users, who are quick with the keyboard, prefer to use access keys instead the mouse.

6. It indicates which character must be pressed in combination with the *Alt* key in order to activate the button from the keyboard. This is standard Windows convention that most users expect.

7. Any button that is frequently clicked should probably be set as a form's default button. This will allow keyboard users to access the button quickly and easily.

8. Because strings are intended to be displayed on a screen or printed on paper. They are stored differently in memory than numbers are.

9. If the number is coded into each statement that uses it, the programmer will have to search through the source code for each instance of the number. If the number is represented by a named constant, however, the programmer need only change the value assigned to the constant.

10. Place the statements in the form's `Load` event handler.

11. Follow these steps:
    1. Select the control you wish to add to the group box.
    2. Cut the control to the clipboard.
    3. Select the group box.
    4. Paste the control.

12. Because language compilers like VB are unable to analyze the logic in a program and follow all the possible execution pathways. If you could create such a tool, you would become very, very rich!

Find the Error

1. The following statement in the `btnSum_Click` event handler is producing the wrong results:

```
sum = CSng(txtNum1.Text & txtNum2.Text & txtNum3.Text)
```

   The statement is concatenating the TextBox controls' Text properties, and the resulting string is being converted to a Single. Correct the problem by converting each TextBox control's Text property to a Single, and then use the + operator to perform addition, as follows:

```
sum = CSng(txtNum1.Text) + CSng(txtNum2.Text) +
      CSng(txtNum3.Text)
```

2. The `btnAverage_Click` event procedure attempts to access the `sum` variable, which is declared in the `btnSum_Click` event procedure. To correct the error, declare a `sum` variable in `btnSum_Click` event procedure and store the sum of the numbers in it.

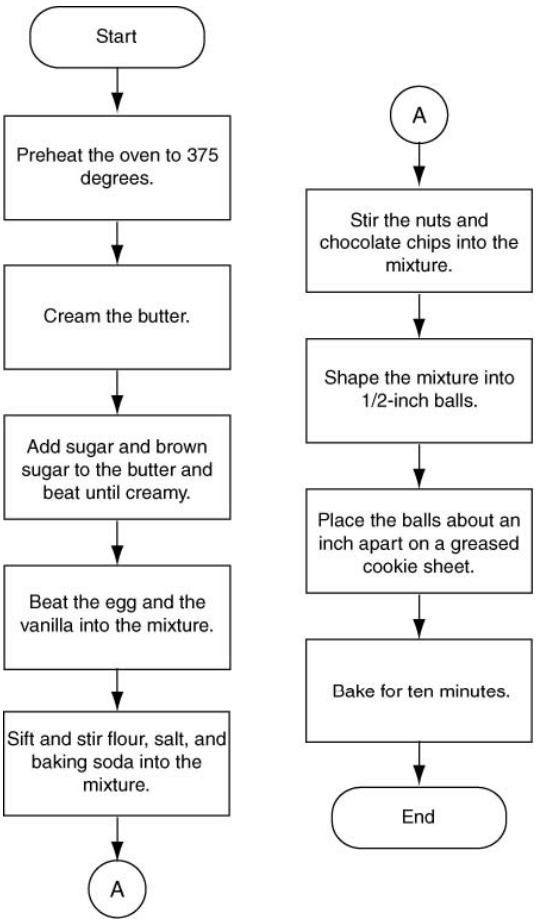
3. This one is TRICKY! As indicated in the remarks, the error lies in the following statement, in the `btnCalculate_Click` event procedure:

```
sum = numberl + number2
```

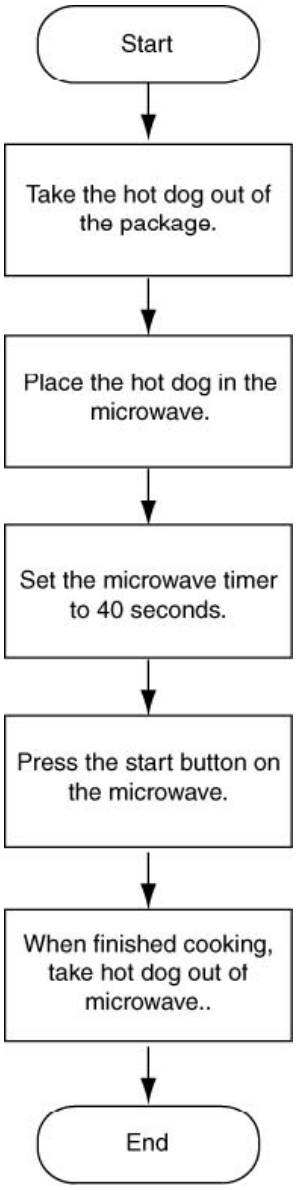
   The programmer has misspelled the name of the variable `number1`. In the statement above, the last character of the variable name is NOT the number one (`1`), but is, in fact, the lowercase letter "L".

Algorithm Workbench

1.



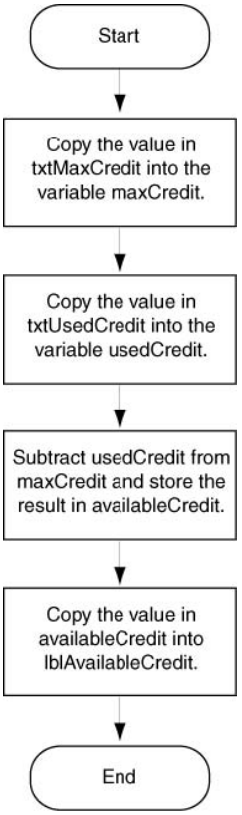
2.



3. The correct order of the statements is:

Assign the value in `txtWidth.Text` *to* `intWidth`.
Assign the value in `txtLength.Text` *to* `intLength`.
Multiply `intWidth` *by* `intLength`, *and store the result in* `intArea`.
Assign the value in `intArea` *to* `lblArea.Text`.

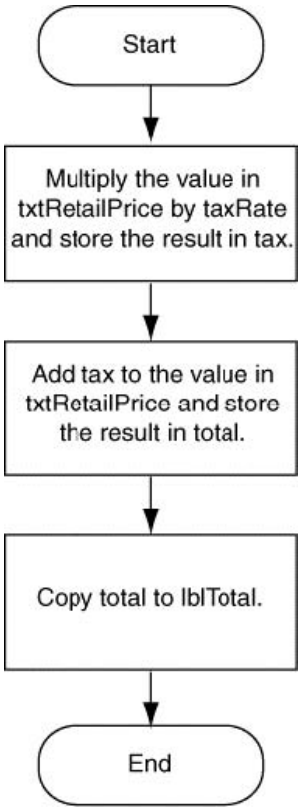
4.



5. Programming statements converted from the flowchart in exercise 4:

a. `decMaxCredit = CDec(txtMaxCredit.Text)`
b. `decUsedCredit = CDec(txtUsedCredit.Text)`
c. `decAvailableCredit = decMaxCredit - decUsedCredit`
d. `lblAvailableCredit.Text = decAvailableCredit.ToString("c")`

6.



Pseudocode:

Multiply the value in `txtRetailPrice` *by* `decTAX_RATE` *and store the result in* `decTax`

Add `decTax` *to the value in* `txtRetailPrice` *and store the result in* `decTotal`

Copy `total` *to* `lblTotal`

7. Programming statements converted from the flowchart or pseudocode in exercise 6:

```
const decTAX_RATE As Decimal = 0.06    ' Tax rate
Dim decTax As Decimal                  ' To hold tax
Dim decTotal As Decimal                ' To hold sale total

' Calculate tax
decTax = CDec(txtRetailPrice.Text) * decTAX_RATE

' Calculate total
decTotal = CDec(txtRetailPrice.Text) + decTax

' Display total
lblTotal.Text = FormatCurrency(decTotal, 2)
```