Chapter 2

1. Revisit the traditional Triangle Program flowchart in Figure 2.1. Can the variable match ever have the value of 4? Of 5? Is it ever possible to "execute" the following sequence of numbered boxes: 1, 2, 5, 6 ?

The values 1, 2, and 3 for variable match indicate which pair of sides are equal. The transitivity of equality means that if any two pairs are equal, the third pair must also be equal. Hence match can only have one of the values 1, 2, 3, or 6. Also, this makes the box sequence 1, 2, 5, 6 is impossible.

2. Recall the discussion from Chapter 1 about the relationship between the specification and the implementation of a program. If you study the implementation of NextDate carefully, you will see a problem. Look at the CASE clause for 30 day months (4, 6, 9, 11). There is no special action for day = 31. Discuss whether or not this implementation is "correct". Repeat this discussion for the treatment of values of day ≥ 29 in the CASE clause for February.

Correctness is a relation between the spec and the code. Since no mention of responses to invalid dates is made in the simple definition of NextDate, the implementation on p. 23 is technically correct. June 31, for example, satisfies the three range definitions. A user would likely not be satisfied with this answer, however. A conscientious developer, upon noticing this problem, would likely ask that the spec be changed. IF not, and the programmer put in a fix that was clearly needed, this would be an example of unspecified, but desirable, functionality.

3. In Chapter 1, we mentioned that part of a test case is the expected output. What would you use as the expected output for a NextDate test case of June 31, 1812? Why?

At a minimum, June 31, 1812 should be identified as an invalid date. The best expected output would be something to the effect that the input date exceeds the maximum day of the month.

4. One common addition to the Triangle Problem is to check for right triangles. Three sides constitute a right triangle if the Pythagorean Relationship is satisfied:

$$c^2 = a^2 + b^2$$

This change makes it convenient to require that the sides be presented in increasing order, i.e., a <= b <= c. Extend the Triangle3 program to include the right triangle feature. We will use this extension in the exercise sections in Parts II and III.

```
'Step 3:  Determine Triangle Type
If IsATriangle
    Then If (a = b) AND (b = c)
            Then Output ("Equilateral")
            ElseIf (a ¹ b) AND (a ¹ c) AND (b ¹ c)
                        Then     Output ("Scalene")
                  delete this Else    Output ("Isosceles")
    replace with Else IF (a^2 = b^2 + c^2) OR (b^2 = a^2 + c^2) OR (c^2 = a^2 + b^2)
                                        Then Output ("Right Triangle")
                                        Else       Output ("Isosceles")
                        EndIf
            EndIf
    ElseOutput("Not a Triangle")
EndIf
```

5. What will the triangle2 program do for the sides -3, -3, 5? Discuss this in terms of the considerations we made in Chapter 1.

It will execute the Output ("Not a Triangle") statement because the c< a + b condition will be false, hence IsATriangle will be false.

6. The function YesterDate is the inverse of NextDate. Given a month, day, year, YesterDate returns the date of
the day before. Develop a program in your favorite language (or our generalized pseudocode) for YesterDate. We will also use this as a continuing exercise.

```
Program YesterDate2
'
Dim yesterDay,yesterMonth,yesterYear As Integer
Dim day,month,year As Integer
Dim c1, c2, c3 As Boolean
'
Do
    Output ("Enter today's date in the form MM DD YYYY")
    Input (month,day,year)
    c1 = (1 <= day) AND (day <= 31)
    c2 = (1 <= month) AND (month <= 12)
    c3 = (1812 <= year) AND (year <= 2012)
    If NOT(c1)
        Then    Output("Value of day not in the range 1..31")
    EndIf
    If NOT(c2)
        Then    Output("Value of month not in the range 1..12")
    EndIf
    If NOT(c3)
        Then    Output("Value of year not in the range 1812..2012")
    EndIf
Until c1 AND c2 AND c2

Case month Of
Case 1: month Is 5, 7, 10, OR 12: 'months after a 30-day month
    If  1< day <= 31
        Then yesterDay = day - 1
        Else
            yesterDay = 30
            yesterMonth = month - 1
    EndIf
Case 2: month Is  2, 4, 6, 8, 9, , Or 11    'months after a 31-day month
    If  1 < day <= 31
        Then    yesterDay = day - 1
        ElseyesterDay = 1
                yesterMonth = month - 1
    EndIf
Case 3: month Is 1: January
    If 1 < day <= 31
        Then yesterDay = day - 1
        Else
            yesterDay = 31
            yesterMonth = 12
```

```
            If year = 2013
                Then Output ("Invalid Input Date")
                Else yester.year = year - 1
    EndIf
Case 4: month is 3: 'March
    If 1 < day <= 31
        Then yesterDay = day - 1
        Else
            If day = 1
                Then
                    If (year is a leap year)
                        Then
                            yesterDay = 29 'leap day
                            yesterMonth = 2
                        Else      'not a leap year
                            yesterDay = 28
                            yesterMonth = 2
                    EndIf
            EndIf
    EndIf
EndCase
Output ("Yesterday's date is", yesterMonth, yesterDay, yesterYear)
'
End YesterDate
```

7. Part of the art of GUI design is to prevent user input errors. Event-driven applications are particularly vulnerable to input errors because events can occur in any order. As the given pseudo-code definition stands, a user could enter a U. S. dollar amount and then click on the compute button without selecting a country. Similarly, could select a country and then click on the compute button without inputting a dollar amount. In an object-oriented application, we can control this by being careful about when we instantiate objects. Revise the GUI class pseudo-code to prevent these two errors.

A reasonable Visual Basic answer is to use the visibility property of the Compute command button. It should be set to "invisible" at design time, and logic associated with the dollar amount and country codes sets it to visible.