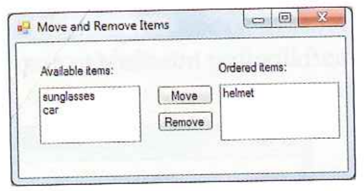
15. Write an event procedure that executes when the user selects an item in `lstStudentName`. The code should determine if the radio button `radRemove` is selected. If it is, the selected list item should be removed from the list box.

16. Write code that adds the value typed in a combo box to the combo box list and then clears the text box of the combo box. Include code that checks the Text property for a valid entry before trying to add the item to the list.

17. An application contains a txtNumSand object, which prompts the user for the number of sandwiches, and a lstSize object, which contains Small, Medium, and Large list items. Write code that executes when a list item is clicked. Include a statement that displays the total cost of the sandwiches in lblCost or the message "Invalid number." The cost of a sandwich is stored in constants named `COST_SMALL`, `COST_MEDIUM`, and `COST_LARGE`.

18. Write a btnCalculate_Click event procedure that adds to lstResult the numbers 1 through 10 and next to each number either their square root rounded to 1 decimal place or their square, depending on the selected radSquareRoots or radSquares radio button. Make the first list item an appropriate title.

19. Use the following interface to answer parts (a) and (b):



a) Write event handlers for each button. The Move button should move a selected item from the Available Items list to the Ordered Items list. The Remove button should move a selected item from the Ordered Items list back to the Available Items list. The buttons should be enabled only when items are available for moving in their respective lists.
b) When the application starts, should there be items in `lstAvailable`, `lstOrdered`, or both? Explain.

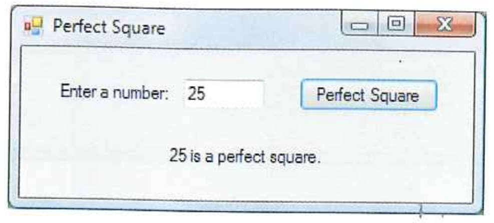
True/False

20. Determine if each of the following are true or false. If false, explain why.
a) The Sqrt() function returns the square root of any number.
b) `IsNumeric(3 - 7)` evaluates to True.
c) `Format(0.079, "Percent")` would display 79%.
d) `Val("$100")` returns 100.
e) The Items.Add() method always adds the new item to the bottom of a list box or combo box.
f) A SelectedIndex value of 0 means no item has been selected in a list box or combo box.
g) The Sorted property of a list box or combo box cannot be set at run time.
h) A Click event is raised when the user selects an item in either a combo box or list box.
i) The Text property of a combo box contains the value of the selected list item.
j) If a list box or combo box cannot display all the items added to the list, a vertical scroll bar is automatically displayed.
k) It is possible to have both Calculate and Choose buttons on the same form.
l) TabIndex values should start at 0.
m) An object with Enabled set to False is not displayed on the form at run time.
n) The trigonometric methods use degree as the angle of measurement.
o) It is possible for certain functions to accept a varying number of arguments.

Exercises

Exercise 1 — PerfectSquare

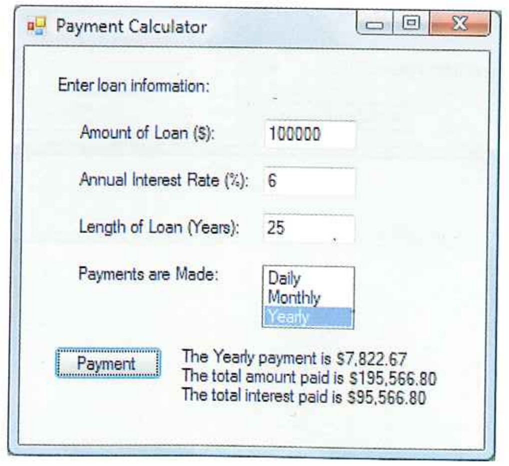
A perfect square is an integer whose square root is a whole number. For example, 4, 9, and 16 are perfect squares. Create a PerfectSquare application that determines if the number entered by the user is a perfect square. The program code should include a PerfectSquare() `Boolean` function that has an `number` parameter. *Hint*: Use the Int() function in determining if a square root is a whole number. The application interface should look similar to:



25 is a perfect square.

Exercise 2 — PaymentCalculator

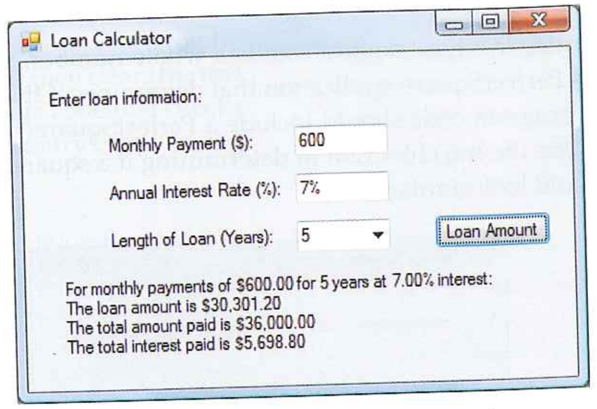
Create a PaymentCalculator application the prompts the user for a loan amount, interest rate, loan term in years, and payments term (daily, monthly, or yearly). Keep in mind the payments must be converted so that the number used in the calculation is in the appropriate units. For this conversion, assume that there are 30 days in each month and there are 360 days in a year. The application should display the payments, total amount paid over the length of the loan, and the total amount of interest paid. The application interface should look similar to:


Enter loan information:
Amount of Loan ($): 100000
Annual Interest Rate (%): 6
Length of Loan (Years): 25
Payments are Made: Daily / Monthly / Yearly

The Yearly payment is $7,822.67
The total amount paid is $195,566.80
The total interest paid is $95,566.80

Exercise 3 — LoanCalculator

Create a LoanCalculator application the prompts the user for the monthly payment, interest rate, and term of a loan in years. The application should display the loan amount, total amount paid, and the total amount of interest paid. The application interface should look similar to:

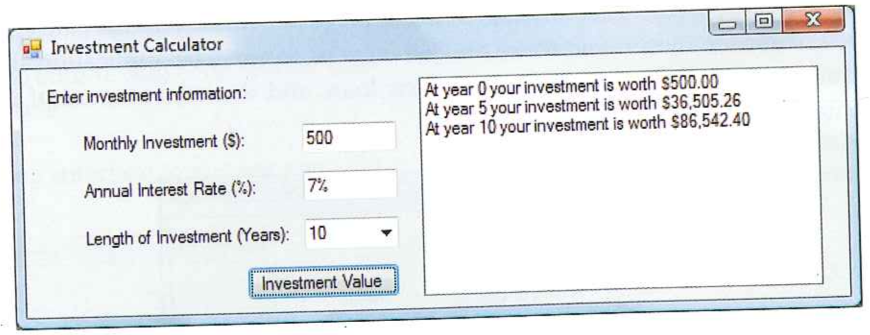


Monthly Payment ($): 600
Annual Interest Rate (%): 7%
Length of Loan (Years): 5

```
For monthly payments of $600.00 for 5 years at 7.00% interest:
The loan amount is $30,301.20
The total amount paid is $36,000.00
The total interest paid is $5,698.80
```

Exercise 4 — InvestmentCalculator

Create an InvestmentCalculator application that prompts the user for the amount invested monthly, interest rate, and term of the investment in years. The application should display the value of the investment at 5 year intervals in a list box and look similar to:

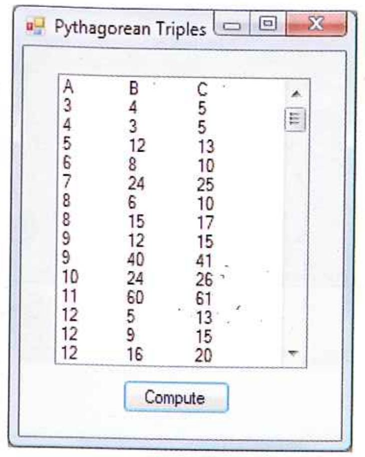

Investment Calculator
Enter investment information:
Monthly Investment ($):
Annual Interest Rate (%):
Length of Investment (Years):
Investment Value

Monthly Investment ($): 500
Annual Interest Rate (%): 7%
Length of Investment (Years): 10

```
At year 0 your investment is worth $500.00
At year 5 your investment is worth $36,505.26
At year 10 your investment is worth $86,542.40
```

Exercise 5 — PythagoreanTriples

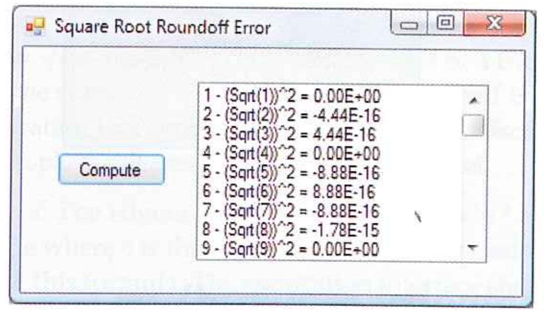
A Pythagorean triple is a set of three integers that solves the equation $a^2 + b^2 = c^2$. Create a PythagoreanTriples application that displays all Pythagorean triples with values of A and B less than 100. The program code should include a PerfectSquare() function like the one created in Exercise 1. The application interface should look similar to the following after clicking Compute:



```
A	B	C
3	4	5
4	3	5
5	12	13
6	8	10
7	24	25
8	6	10
8	15	17
9	12	15
9	40	41
10	24	26
11	60	61
12	5	13
12	9	15
12	16	20
```

Exercise 6 — SquareRootRoundoffError

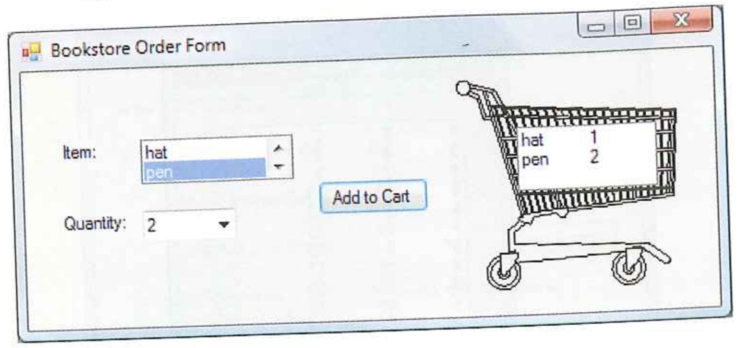
Create a SquareRootRoundoff Error application that compares the square of the square root of a number with the number itself for all integers from 1 to 100. This difference in values is due to the computer's rounding error. The application interface should look similar to the following after clicking Compute:



```
1 - (Sqrt(1))^2 = 0.00E+00
2 - (Sqrt(2))^2 = -4.44E-16
3 - (Sqrt(3))^2 = 4.44E-16
4 - (Sqrt(4))^2 = 0.00E+00
5 - (Sqrt(5))^2 = -8.88E-16
6 - (Sqrt(6))^2 = 8.88E-16
7 - (Sqrt(7))^2 = -8.88E-16
8 - (Sqrt(8))^2 = -1.78E-15
9 - (Sqrt(9))^2 = 0.00E+00
```

Exercise 7 — BookstoreOrderForm

Create a BookstoreOrderForm application the prompts the user to select different items and quantities to purchase and then displays the order information in a list box that is on top of a `cart.gif` graphic, a data file for this text. The application interface should look similar to:

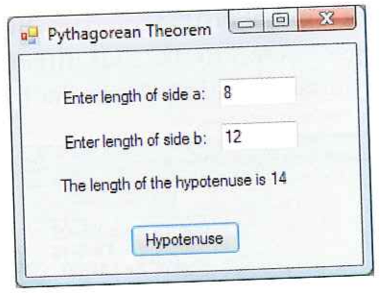


Item: hat, pen
Quantity: 2

hat	1
pen	2

Exercise 8 — PythagoreanTheorem

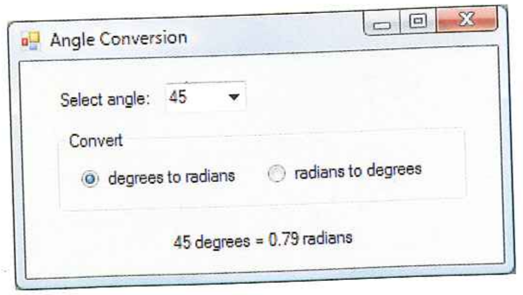
The Pythagorean Theorem is $a^2 + b^2 = c^2$ where a and b are the lengths of two sides of a right triangle and c is the length of the side opposite the right angle (the hypotenuse). Create a PythagoreanTheorem application that prompts the user for the lengths of sides `a` and `b` and then displays the length of the hypotenuse when **Hypotenuse** is clicked. The application interface should look similar to:



Enter length of side a: 8
Enter length of side b: 12
The length of the hypotenuse is 14

Exercise 9 (trigonometry required) — AngleConversion

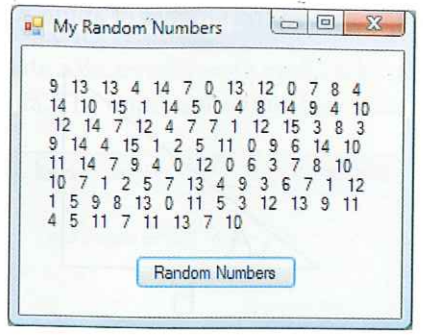
Create an AngleConversion application that converts an angle in degrees to radians and vice versa. The application should include the options 30, 0.52, 45, 0.79, 60, and 1.05 in a combo box. Refer to the functions created in the reviews for this chapter. The application interface should look similar to:



Select angle: 45
[✓ degrees to radians] radians to degrees
45 degrees = 0.79 radians

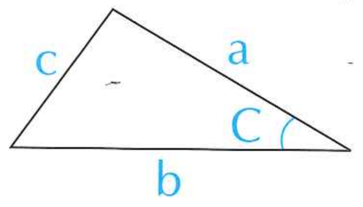
Exercise 10 (trigonometry required) — MyRandomNumbers

Create a MyRandomNumbers application that produces a sequence of random numbers without using the Rnd() function. To do this, let `X` vary from 1 to 100 in steps of 1. Obtain Sin(`X`) and multiply this by `1000`, which results in a value `Y`. Then take the absolute value of `Y` and divide `Int(Y)` by `16`, and let the remainder serve as the random number. The application interface should look similar to the following after clicking **Random Numbers**:



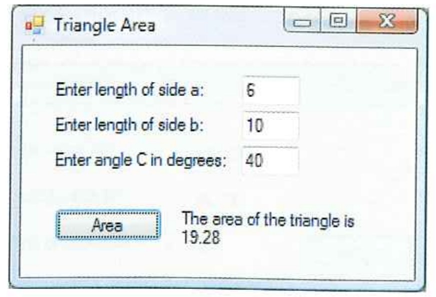
```
9 13 13 4 14 7 0 13 12 0 7 8 4
14 10 15 1 14 5 0 4 8 14 9 4 10
12 14 7 12 4 7 7 1 12 15 3 8 3
9 14 4 15 1 2 5 11 0 9 6 14 10
11 14 7 9 4 0 12 0 6 3 7 8 10
10 7 1 2 5 7 13 4 9 3 6 7 1 12
1 5 9 8 13 0 11 5 3 12 13 9 11
4 5 11 7 11 13 7 10
```

Exercise 11 — TriangleArea



a) The formula Area = $\sqrt{s(s-a)(s-b)(s-c)}$ computes the area of a triangle where `a`, `b`, and `c` are the lengths of the sides and `s` is the semiperimeter (half the perimeter). Create a TriangleArea application that prompts the user with text boxes for lengths of sides `a`, `b`, and `c` and then displays the area of the triangle in a label.

b) *Trigonometry required.* The trigonometric formula Area = ½ * a * b * Sin C computes the area of a triangle where `C` is the angle formed by sides `a` and `b`, in degrees. Modify TriangleArea to use this formula. The application interface should look similar to:



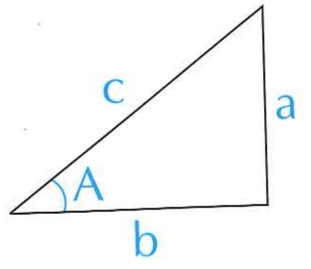
Enter length of side a: 6
Enter length of side b: 10
Enter angle C in degrees: 40
The area of the triangle is 19.28

Exercise 12 (trigonometry required) — SOH–CAH–TOA

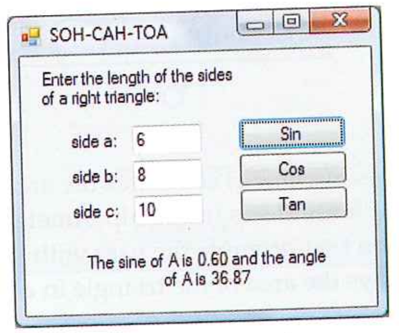
SOH-CAH-TOA is a mnemonic for the trigonometric formulas that can be used to find the sine, cosine, and tangent of an angle in a right triangle:

Sine = Opposite/Hypotenuse Cosine = Adjacent/Hypotenuse Tangent = Opposite/Adjacent

For example, the sine of angle A is calculated by dividing the opposite side a by the hypotenuse c. Likewise, the cosine of angle A is b/c, and the tangent of angle A is a/b:



Create a SOH-CAH-TOA application that prompts the user for the lengths of the three sides of a triangle and then calculates the sine, cosine, and tangent of angle A depending on which button is clicked. The application should use the RadiansToDegrees() function from the reviews in this chapter to determine angle A in degrees. The code should also include a CheckSides() `Boolean` function that checks that the sides entered form a right triangle (Hint: Refer to Exercise 8) and if not, displays a message box and clears the text boxes to allow for new entries. The application interface should look similar to:



Enter the length of the sides of a right triangle:

side a: 6
side b: 8
side c: 10

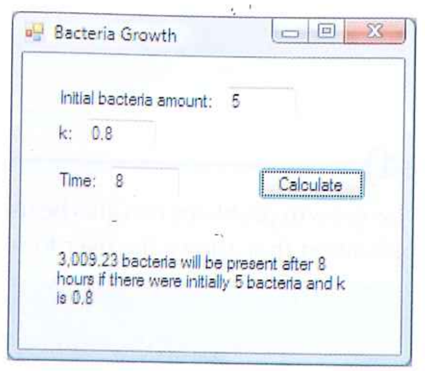
The sine of A is 0.60 and the angle of A is 36.87

Exercise 13 — BacteriaGrowth

The formula y = ne^{kt} can be used for estimating growth where:

y	is the final amount
n	is the initial amount
k	is a constant
t	is the time

For example, this formula could be used for estimating population growth in a region or for estimating cell growth in a lab experiment. Create a BacteriaGrowth application that calculates how many bacteria will be present based on this formula. The application interface should look similar to:



Initial bacteria amount: 5
k: 0.8
Time: 8

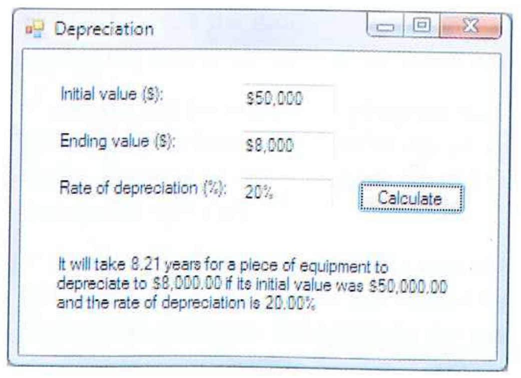
3,009.23 bacteria will be present after 8 hours if there were initially 5 bacteria and k is 0.8

Exercise 14 — Depreciation

The formula $V_n = P(1 + r)^n$ is can be used to estimate depreciation or appreciation where:

V_n	is the value at the end of n years
P	is the initial value of the equipment
r	is the rate of appreciation (positive) or depreciation (negative)
n	is the time in years

For example, this formula could be used to determine the current value of a mainframe that a company has owned for 10 years. From this formula you can also determine how long it will take a piece of equipment to depreciate to a specific value using the formula: $n = \log(V_n / P) / \log(1 + r)$. Create a Depreciation application that calculates how long it will take a piece of equipment to depreciate using this formula. The application interface should look similar to:



Initial value ($): $50,000
Ending value ($): $8,000
Rate of depreciation (%): 20%

It will take 8.21 years for a piece of equipment to depreciate to $8,000.00 if its initial value was $50,000.00 and the rate of depreciation is 20.00%

Exercise 15 (trigonometry required) (advanced) —— Triangle

Create an application that solves a triangle (compute the unknown sides and angles) for the following situations:

- given two sides and the included angle
- given two angles and any side
- given three sides

The application should prompt the user to select one of the three choices, and based on the selected option prompts the user to enter the appropriate known information. Angles should be entered in degrees and displayed in degrees.

Exercise 16 (advanced) —— Decay

The formula used in Exercise 13 for growth problems can also be used in decay problems. In decay problems, `k` is negative. Create an application that allows the user to select from the following options:

- calculate the final amount `y`: ne^{-kt}
- calculate the initial amount `n`: y / e^{-kt}
- calculate the constant `k` (called the half-life): `(log (y/n)) / t`

The application should prompt the user to select one of the three choices and based on the selected option prompts the user to enter the appropriate known information. For example, a radioactive mass of 200 grams will reduce to 100 grams in 10 years. Based on this information, the half-life is calculated to be –0.06931.