17. Write code that prompts the user for a date in the form mm/dd/yy and then displays the date in form month, day, year. For example, 12/25/13 displays December 25, 2013.

18 What is displayed by the message box?

```
nums = "01234567890123456789012345678890"
MessageBox.Show(nums & vbCrLf & vBTab &
"This" & vbTab & "Is" & vbCrLf &
"So Wonderful" & "and Marvelous")
```

19. What is displayed in the label?

```
original = "Visual Basic is so much fun!"
length = original.Length
For i As Integer = 0 To length - 1
    char = original.Substring(i, 1)
    If i Mod 2 = 0 Then
        output = output & char.ToUpper
    Else
        output = output & char.ToLower
    End If
Next i
Me.lblOutput.Text = output
```

20. a) Write an algorithm to count the number of words in a sentence.
    b) Write an algorithm to count the number of letters in a sentence.

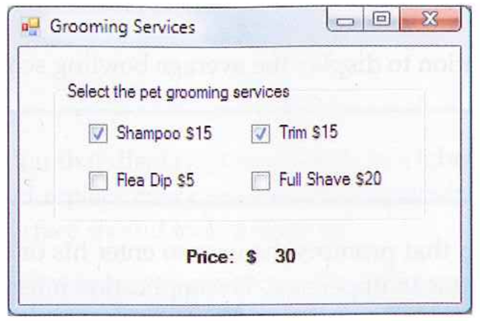
True/False

21. Determine if each of the following is true or false. If false, explain why.
    a) Each execution of a loop is called an iteration.
    b) A `Do...Loop` that evaluates the condition before executing a loop may never execute.
    c) An infinite loop continues forever.
    d) Accumulator variables can only be Integer variables.
    e) An accumulator signifies that a loop is to stop iterating.
    f) Sentinel values must always be the value –1.
    g) The statement `sum += 1` uses `sum` as an accumulator variable.
    h) The counter in a `For...Next` statement is always incremented or decremented by 1.
    i) The `String` data type is a primitive data type.
    j) A string can be changed by methods of the `String` class.
    k) `String.Compare("Test", "TEST", False)` returns a negative value.
    l) `vbTab` is a variable that allows the programmer to control the spaces in a string.
    m) The Unicode value for an uppercase letter is the same for the corresponding lowercase letter.
    n) The Compare() method returns a `Boolean` variable.
    o) A wildcard character is a ?, *, or #, which matches an unknown character or group of characters.

Exercises

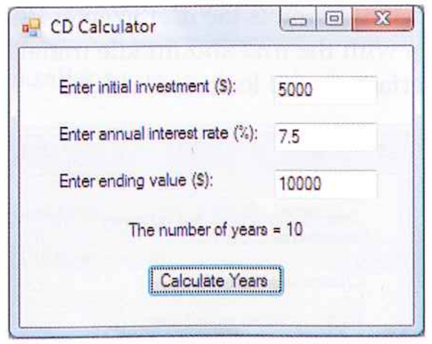
Exercise 1 — GroomingServices

Create an GroomingServices application that allows the user to select pet grooming services. A shampoo is $15, a flea dip is $5, a trim is $15, and a full shave is $20. The displayed price should reflect the total of the services currently selected. If a service is cleared, the price should change immediately to reflect the correct price. *Hint*: Use Click event procedures for each check box to update an accumulator. The application interface should look similar to the following after selecting the Shampoo and Trim check boxes:


Grooming Services
Select the pet grooming services
☑ Shampoo $15 ☑ Trim $15
☐ Flea Dip $5 ☐ Full Shave $20
Price: $ 30

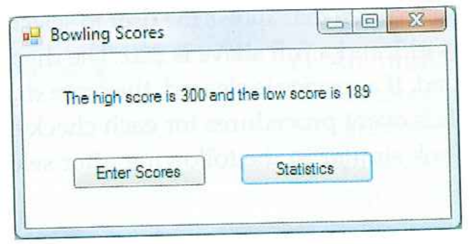
Exercise 2 — CDCalculator

A certificate of deposit (CD) is a type of investment that matures at a specified interest rate for a specified period. Create a CDCalculator application that prompts the user for the initial investment amount, the annual interest rate, and the desired ending value and then displays the number of years required for the CD to be worth the specified ending value when interest is compounded annually. The CD value at the end of each year can be calculated by the formula CD Value = CD Value + (CD Value * Interest Rate). To determine the number of years it will take for the CD to reach the desired ending value, repeatedly execute the formula until the CD Value is greater than or equal to the desired ending value. The application interface should look similar to:


CD Calculator
Enter initial investment ($): 5000
Enter annual interest rate (%): 7.5
Enter ending value ($): 10000
The number of years = 10

Exercise 3 — BowlingScores

a) Create a BowlingScores application that prompts the user to enter as many bowling scores as desired and then displays the high score and the low score. The application interface should look similar to the following after clicking Enter Scores, entering a set of scores, and then clicking Statistics:

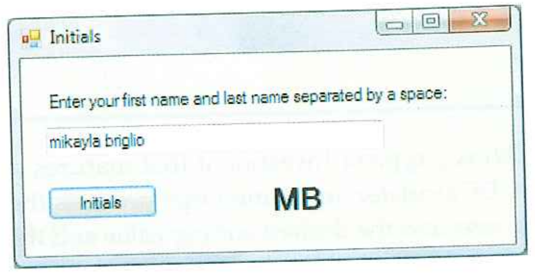


The high score is 300 and the low score is 189

b) Modify the application to display the average bowling score in a label.

Exercise 4 — Initials

Create an Initials application that prompts the user to enter his or her first and last names and then displays the initials of the name in uppercase. The application interface should look similar to:



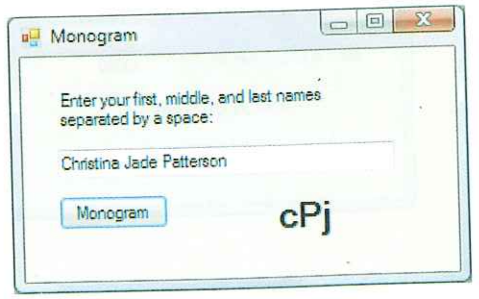
Enter your first name and last name separated by a space:

mikayla briglio

MB

Exercise 5 — Monogram

Create a Monogram application that prompts the user to enter his or her first, middle, and last names and then displays a monogram with the first and middle initials in lowercase and the last initial in uppercase. The application interface should look similar to:



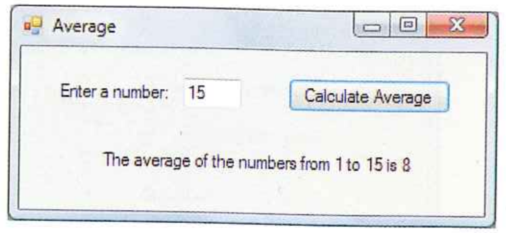
Enter your first, middle, and last names separated by a space:

Christina Jade Patterson

cPj

Exercise 6 — Average

Create an Average application that calculates the average of a set of numbers from 1 to a number entered by the user. For example, if the user enters 5, the average of 1, 2, 3, 4, and 5 would be calculated. The application interface should looks similar to:

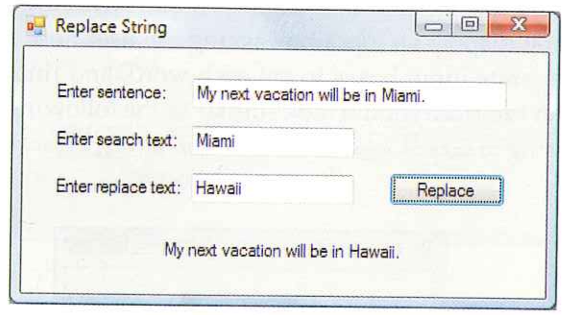


Enter a number: 15

The average of the numbers from 1 to 15 is 8

Exercise 7 — ReplaceString

Create a ReplaceString application that displays a new string in a label. The new string should take a sentence entered by the user and replace every occurrence of a substring with a new string supplied by the user. The application interface should look similar to:



Enter sentence: My next vacation will be in Miami.

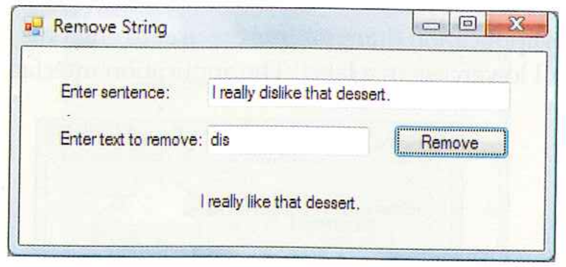
Enter search text: Miami

Enter replace text: Hawaii

My next vacation will be in Hawaii.

Exercise 8 — RemoveString

Create a RemoveString application that displays a new string in a label. The new string should take a sentence entered by the user and remove every occurrence of a substring supplied by the user. The application interface should look similar to:



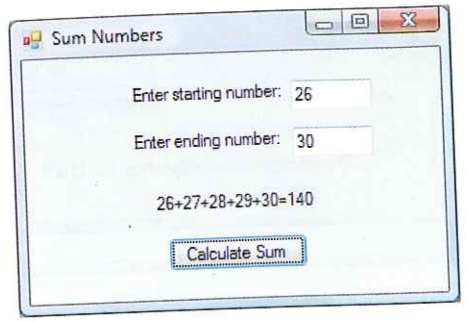
Enter sentence: I really dislike that dessert.

Enter text to remove: dis

I really like that dessert.

Exercise 9 SumNumbers

Create a SumNumbers application that calculates the sum of a range of numbers entered by the user and displays in a label an expression with the numbers in the range. The application interface should look similar to:


Sum Numbers

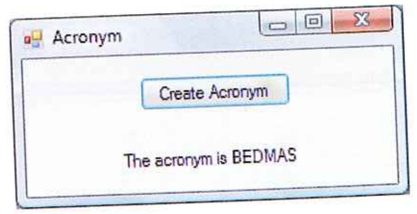
Enter starting number: 26

Enter ending number: 30

26+27+28+29+30=140

Exercise 10 Acronym

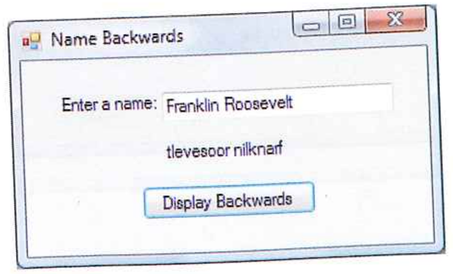
An acronym is a word formed from the first letters of a few words, such as GUI for graphical user interface. Create an Acronym application that displays an acronym for the words entered by the user. The application should first display an input box asking the user how many words will make up the acronym, then display separate input boxes to get each word, and finally display the acronym in all uppercase. The application interface should look similar to the following after clicking Create Acronym, entering 6, and then entering `brackets exponents division multiplication addition subtraction` as the words:


Acronym

The acronym is BEDMAS

Exercise 11 NameBackwards

As a young boy Franklin Roosevelt signed his letters to his mother backwards: tlevesoor nilknarf. Create a NameBackwards application that prompts the user to enter his or her name and then displays the name backwards in all lowercase in a label. The application interface should look similar to:

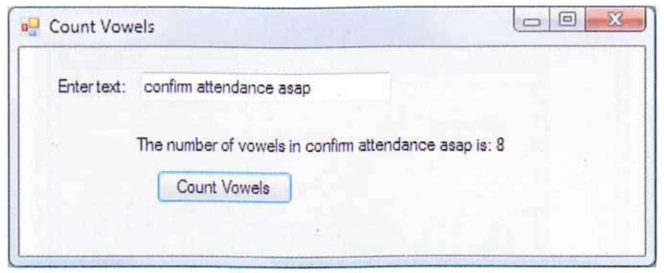

Name Backwards

Enter a name: Franklin Roosevelt

tlevesoor nilknarf

Exercise 12 CountVowels

Create a CountVowels application that counts the number of vowels in a word or phrase. The application interface should look similar to:

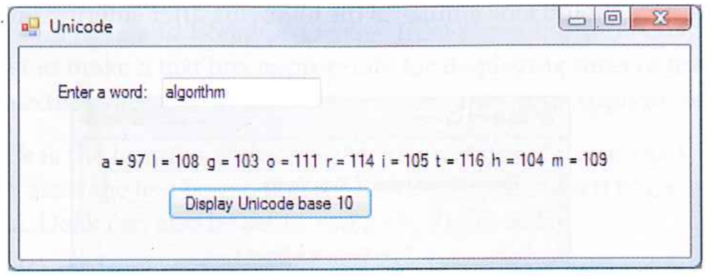

Count Vowels

Enter text: confirm attendance asap

The number of vowels in confirm attendance asap is: 8

Exercise 13 Unicode

Create a Unicode application that prompts the user for a word and then displays the Unicode base 10 number for each letter in the word. The application interface should look similar to:

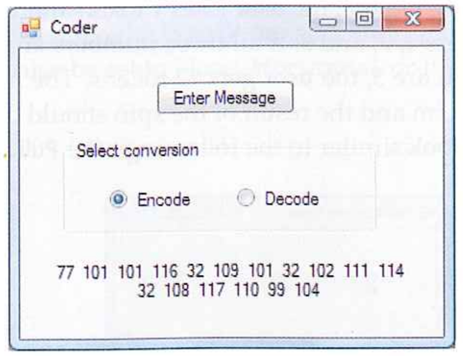

Unicode

Enter a word: algorithm

```
a = 97 l = 108 g = 103 o = 111 r = 114 i = 105 t = 116 h = 104 m = 109
```

Exercise 14 Coder

a) Create a Coder application that encodes or decodes a message using Unicode. The application interface should look similar to the following after clicking Enter Message, entering `Meet me for lunch` in an input box, and then selecting Encode:


Coder

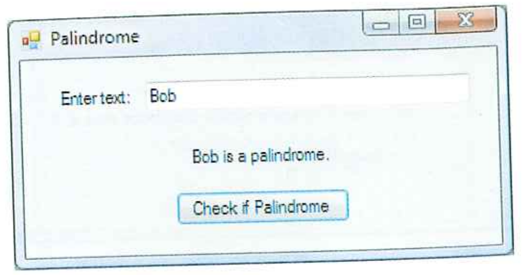
Select conversion: Encode [✓] Decode

```
77 101 101 116 32 109 101 32 102 111 114
32 108 117 110 99 104
```

b) Modify Coder to produce the code by converting each letter in the original message to its corresponding Unicode number, add 2 to each number, and then convert these numbers back to characters to display a coded message. Keep all spaces between the words in their original places and realize that the letters "Y" and "Z" are to be converted to A and B.

Exercise 15 — Palindrome

A palindrome is a word or phrase that is spelled the same backwards and forwards, such as madam, dad, or race car. Create a Palindrome application that uses a loop to determine if the word or phrase entered by the user is a palindrome. The application interface should look similar to:

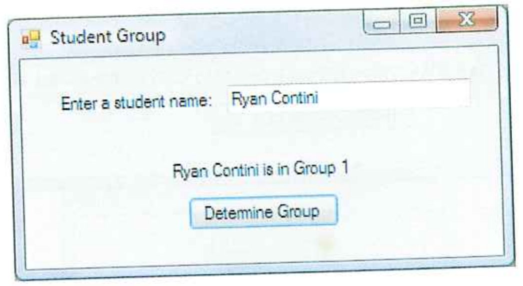


Enter text: Bob

Bob is a palindrome.

Exercise 16 — StudentGroup

Create a StudentGroup application that prompts the user to enter a student name and then displays what group a student is assigned to depending on the first letter in the student's last name. Last names beginning with A through I are in Group 1, J through S are in Group 2, T through Z are in Group 3. The application interface should look similar to the following after entering text and clicking Determine Group:

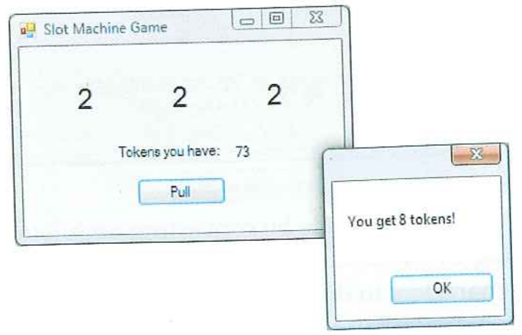


Enter a student name: Ryan Contini

Ryan Contini is in Group 1

Exercise 17 — SlotMachineGame

Create a SlotMachineGame application that acts as a simple slot machine. The user starts with 100 tokens. With each "pull" of the handle, the user loses 1 token and the computer "spins" three wheels, each consisting of the numbers 1, 2, and 3. If all three numbers are 1, the user gets 4 tokens; if all are 2, the user gets 8 tokens; if all are 3, the user gets 12 tokens. The number of tokens that the user has should be displayed on the form and the result of the spin should be displayed in a message box. The application interface should look similar to the following after Pull has been clicked several times:



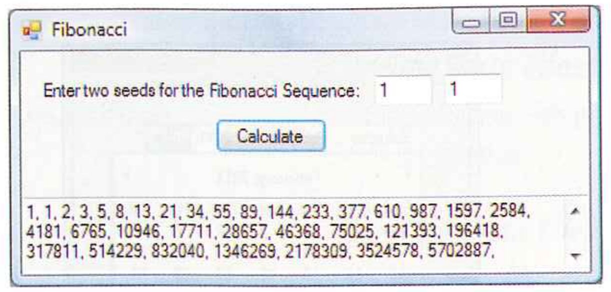
2 2 2

Tokens you have: 73

You get 8 tokens!

Exercise 18 — Fibonacci

Create a Fibonacci application that gets two seeds from the user and then generates a Fibonacci Sequence of 50 values:



Enter two seeds for the Fibonacci Sequence: 1 1

```
1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584,
4181, 6765, 10946, 17711, 28657, 46368, 75025, 121393, 196418,
317811, 514229, 832040, 1346269, 2178309, 3524578, 5702887,
```

A *Fibonacci Sequence* is a sequence of numbers that is generated from two starting values called seeds. The two seeds are the first two numbers in the sequence. The seeds are then added together to generate the third number in the sequence. Each additional number in the sequence is created by adding the previous two numbers. For example, if the seeds are 1 and 1, the sequence is 1, 1, 2, 3, 5, 8, ... As another example, the seeds 3 and 6 generate the sequence 3, 6, 9, 15, 24, 39, ...

A long series of numbers will need to be displayed. A label is not a good choice for such a long list of numbers. TextBox objects have additional properties from those listed in Chapter 3. The properties listed below can be set to make a text box appropriate for displaying lines of text, such as a sequence of numbers. Use the TextBox properties listed below to enhance your application:

- **Dock** is the location of docking for the text box. Setting Dock to Bottom automatically sizes the text box so that the bottom, right, and left borders are anchored to the form. Dock can also be set to Top, Left, Right, or Fill.
- **ReadOnly** is set to True when the text box should be used for display only. When ReadOnly is True, text can be displayed in the text box, but the user will not be able to type in the text box.
- **Multiline** is set to True to allow the text box to display multiple lines of text. The box can then be sized vertically.
- **WordWrap** is set to True to wrap lines of text at the right border of the text box. WordWrap can be set to False, but it is not recommended because only one line of text will be displayed and it may extend beyond the right border of the text box.
- **ScrollBars** is set to Vertical to add a vertical scroll bar to the text box. A vertical scroll bar appears only when the WordWrap and Multiline properties are also set to True. ScrollBars can also be set to None, Horizontal, or Both.