Math Operators

Click on the following link for a table of Math Operators as well as a description of their uses.

Overview of Math Operators

The Change Machine

Imagine going to the store and purchasing something for $3.26. You hand the cashier a twenty dollar bill and the hand you back your change in nickels, dimes and pennies.

If that cashier had had our handy, dandy ‘Change Machine”, that would never have happened!

Imagine the cashier till had the following currencies:

twenty dollar bills
ten dollar bills
five dollar bills
toonies
loonies
quarters
dimes
nickels
pennies

Question: how many twenty dollar bills will be part of the change and how much will be left over after the twenties have been dispensed if $58.26 is the amount of change being returned.

How Many Twenties?

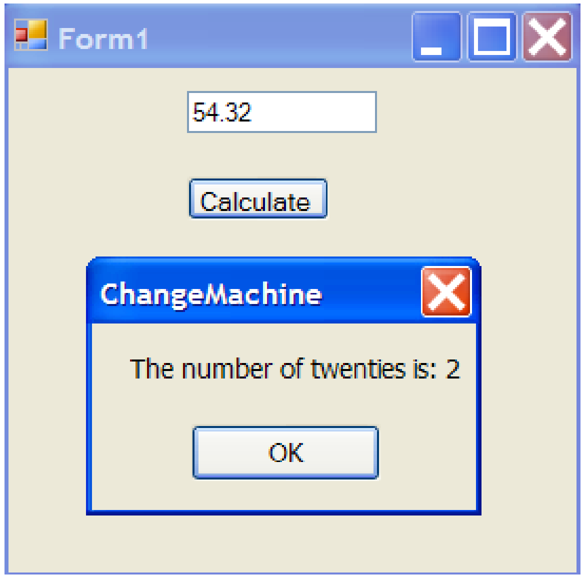
How many twenty dollar bills will be part of $54.32 in change

Dividing 54.32 gives us an answer of 2.716. It is pretty difficult to hand out 2.716 twenty dollar bills.

If the variable that accepted the answer of 54.32 / 20 was an integer variable we would get a nice even number

```
Private Sub btnCalculate_Click(ByVal sender As System.Object,
    Dim numTwenties As Integer
    Dim change As Decimal
    change = Val(txtChange.Text)
    numTwenties = change \ 20
    MsgBox("The number of twenties is:  " & numTwenties)
End Sub
```

Integer division operator
is a backslash



The number of twenties is: 2

What…More Change?

While I’m sure the customer in question will be pleased to see the 2 twenties being returned in change, they are not likely to be too impressed if that is all they get.

So far we have seen $40 of the $54.32 expected.

In order to keep track of the remainder and to determine what currencies will be necessary we need to use the modulus operator.

The modulus operator is simply the word Mod in Visual Basic.

This operation is concerned with what is left over after the integer division

For example: 8 Mod 3.

We know that 3 will go into 8, 2 times evenly. This accounts for the a value of 6 but there is still 2 left over. Therefore the answer to 8 Mod 3 is 2

Practice with the Modulus Operator

8 Mod 5	3
8 Mod 6	2
8 Mod 7	1
8 Mod 4	0
8 Mod 2	0
10 Mod 3	1
10 Mod 5	0
21 Mod 2	1
21 Mod 7	0
100 Mod 3	2
99 Mod 3	0
88 Mod 22	0
88 Mod 12	4

Remainders From Modulus

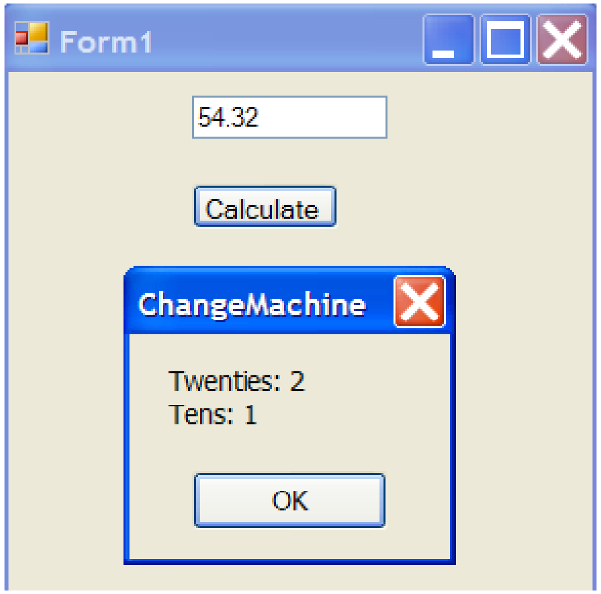
Consider our change of $54.32 again.

54.32 \ 20(integer division) will tell us the number of twenties will fit evenly into 54.32.

54 Mod 20 will tell us the remainder after the twenties have been issud.

The remainder that we have can now be used to decide what size currencies should be used for the rest of the change.

```
change = Val(txtChange.Text)
numTwenties = change \ 20
remainder = change Mod 20
numTens = remainder \ 10
MsgBox("Twenties: " & numTwenties & wrap & "Tens: " & numTens)
```



Twenties: 2
Tens: 1

How Many Tens?

The amount of change left over after the twenty dollar bills have been issues is
$14.32 (54.32 – 40).

We calculate this by writing change Mod 20 and assign the answer to a variable we will call remainder.

We can now use this value to calculate the number of ten dollar bills by writing
numTens = remainder \ 10

The variable remainder will have a constantly changing and diminishing value as we hand out more currencies.

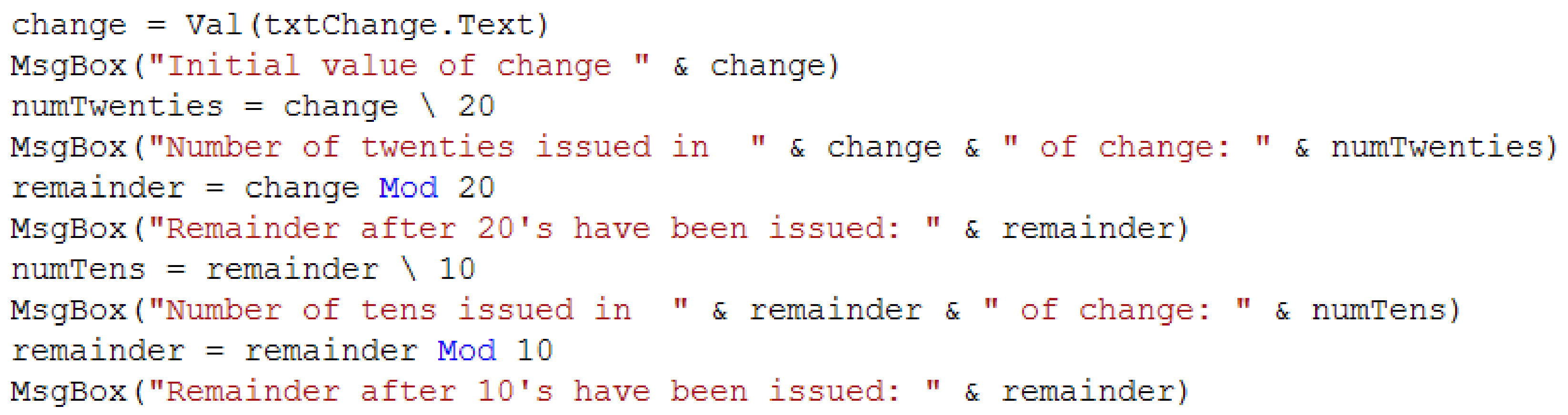
After we hand out the ten dollar bills we can calculate the new value of remainder by writing:

remainder = remainder Mod 10

Inserting Message Boxes to Track Values

As you progress through a program like this it can become easy to lose track of the values held by the variables. One trick you can use is to insert message boxes that flash the various values as the program is run.

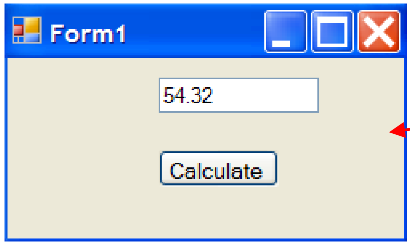
After your program is completed and tested you can remove these extra message boxes.



```
change = Val(txtChange.Text)
MsgBox("Initial value of change " & change)
numTwenties = change \ 20
MsgBox("Number of twenties issued in  " & change & " of change: " & numTwenties)
remainder = change Mod 20
MsgBox("Remainder after 20's have been issued: " & remainder)
numTens = remainder \ 10
MsgBox("Number of tens issued in  " & remainder & " of change: " & numTens)
remainder = remainder Mod 10
MsgBox("Remainder after 10's have been issued: " & remainder)
```

There are a lot of message boxes in the above example even for a testing draft of your program. More likely is that you would track values through message boxes only in complicated sections of code that were giving you strange value errors.

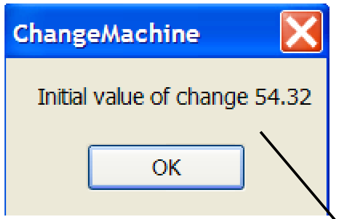
Tracking Values


Form1

54.32

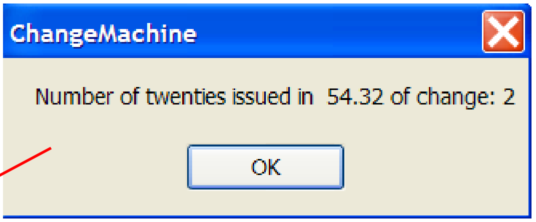
Calculate

Initial Form


ChangeMachine

Initial value of change 54.32

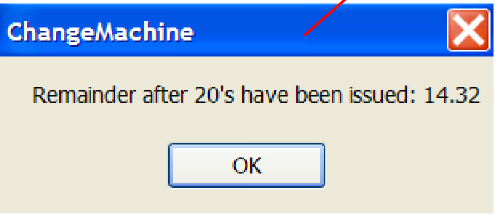
OK


ChangeMachine

Number of twenties issued in 54.32 of change: 2

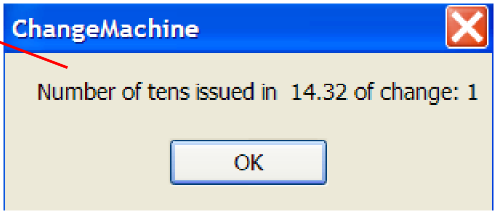
OK

```
change = Val(txtChange.Text)
MsgBox("Initial value of change " & change)
numTwenties = change \ 20
MsgBox("Number of twenties issued in  " & change & " of change: " & numTwenties)
remainder = change Mod 20
MsgBox("Remainder after 20's have been issued: " & remainder)
numTens = remainder \ 10
MsgBox("Number of tens issued in  " & remainder & " of change: " & numTens)
remainder = remainder Mod 10
MsgBox("Remainder after 10's have been issued: " & remainder)
```


ChangeMachine

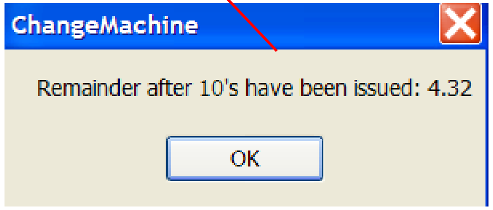
Remainder after 20's have been issued: 14.32

OK


ChangeMachine

Number of tens issued in 14.32 of change: 1

OK


ChangeMachine

Remainder after 10's have been issued: 4.32

OK

Divide and Conquer

Some additional tips for larger programs.

If you have a large program, concentrate on getting smaller components of the program working first.

Don’t write 100 limes of code before you test and find out that everything you did after line 6 was in the wrong direction.

In the ChangeMachine, concentrate on getting the number of 20’s issue correct along with the remainder. If you can solve the problem for 20’s, most of your work is already done for the remainder of the currencies. In fact, you can copy and paste a lot of the rest of the program.

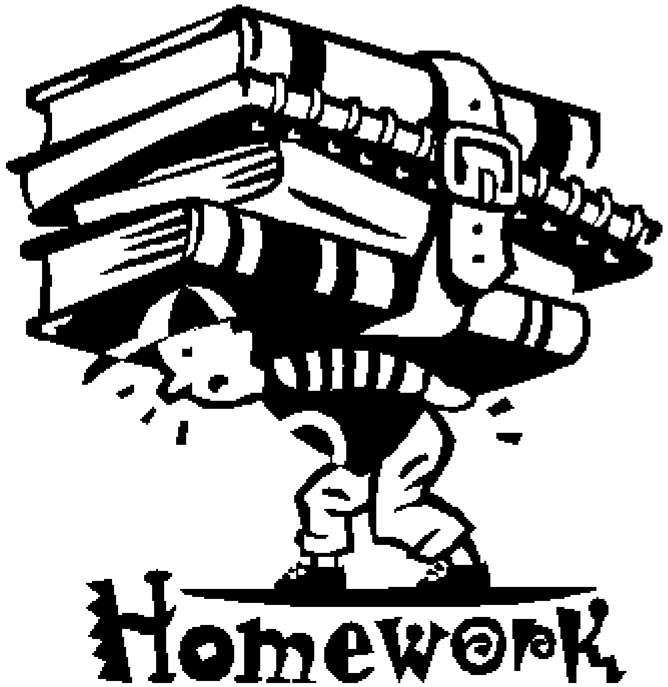
Draw diagrams to chart the flow of the program. A lot of people are able to understand a problem a lot better when it is laid out for them visually.

Comment your code. Writing down the meaning of the individual lines can go a long way to reinforcing your understanding and memory of the code.

Create Mini-Programs that model the problems you encounter in bigger programs

Big Problems can be reduced to a series of little problems. Little problems can be broken down into a series of simple steps.

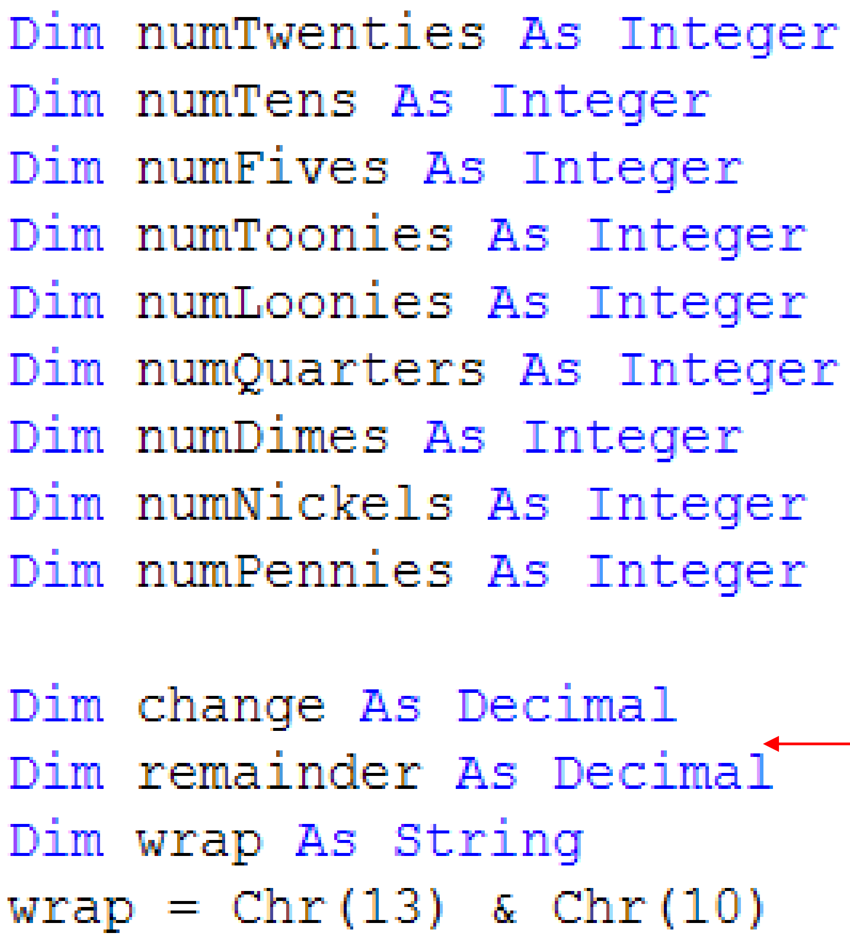
Complete The Change



Finish the program so that it calculates and displays the number of currencies from twenty dollar bills down to pennies.

Comment the program thoroughly.

The Solution…..for shameless cheaters!



```
Dim numTwenties As Integer
Dim numTens As Integer
Dim numFives As Integer
Dim numToonies As Integer
Dim numLoonies As Integer
Dim numQuarters As Integer
Dim numDimes As Integer
Dim numNickels As Integer
Dim numPennies As Integer

Dim change As Decimal
Dim remainder As Decimal
Dim wrap As String
wrap = Chr(13) & Chr(10)
```

The declarations

Using the Decimal variable type when dealing with money.

Twenties to Loonies

```
change = Val(txtChange.Text)
MsgBox("Initial value of change " & change)
numTwenties = change \ 20
MsgBox("Number of twenties issued in  " & change & " of change: " & numTwenties)
remainder = change Mod 20
MsgBox("Remainder after 20's have been issued: " & remainder)

numTens = remainder \ 10
MsgBox("Number of tens issued in  " & remainder & " of change: " & numTens)
remainder = remainder Mod 10
MsgBox("Remainder after 10's have been issued: " & remainder)

numFives = remainder \ 5
MsgBox("Number of fives issued in  " & remainder & " of change: " & numFives)
remainder = remainder Mod 5
MsgBox("Remainder after 5's have been issued: " & remainder)

numToonies = remainder \ 2
MsgBox("Number of toonies issued in  " & remainder & " of change: " & numToonies)
remainder = remainder Mod 2
MsgBox("Remainder after toonies have been issued: " & remainder)

numLoonies = remainder \ 1
MsgBox("Number of loonies issued in  " & remainder & " of change: " & numLoonies)
remainder = remainder Mod 1
MsgBox("Remainder after loonies have been issued: " & remainder)
```

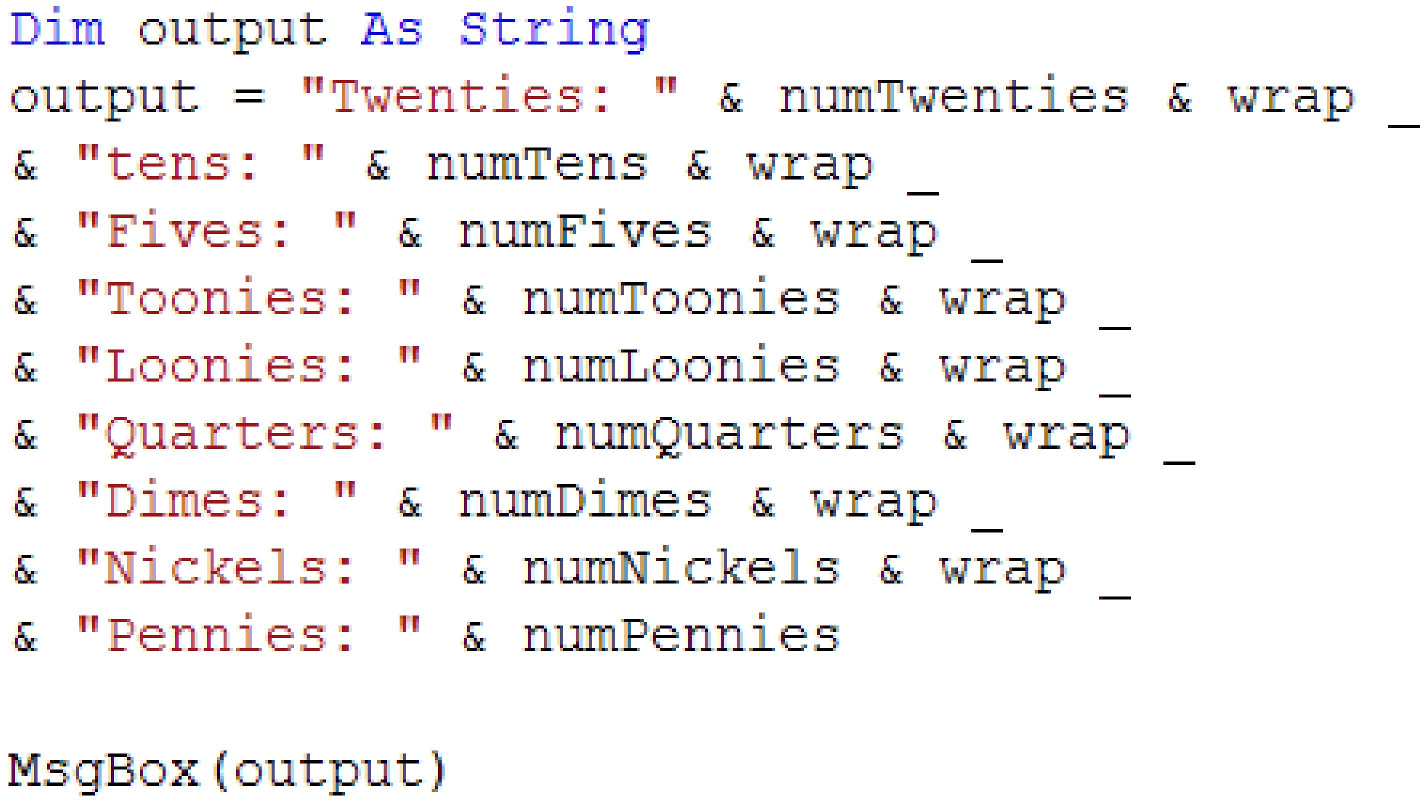
Quarters to Pennies

When dealing with Integer division avoid use decimals, thus the X 100

```
numQuarters = (remainder * 100) \ 25
MsgBox("Number of quarters issued in  " & remainder & " of change: " & numQuarters)
remainder = remainder Mod 0.25
MsgBox("Remainder after quarters have been issued: " & remainder)

numDimes = remainder * 100 \ 10
MsgBox("Number of dimes issued in  " & remainder & " of change: " & numDimes)
remainder = remainder Mod 0.1
MsgBox("Remainder after dimes have been issued: " & remainder)

numNickels = remainder * 100 \ 5
MsgBox("Number of nickels issued in  " & remainder & " of change: " & numNickels)
remainder = remainder Mod 0.05
MsgBox("Remainder after nickels have been issued: " & remainder)

numPennies = remainder * 100 \ 1
MsgBox("Number of pennies issued in  " & remainder & " of change: " & numPennies)
remainder = remainder Mod 0.01
MsgBox("Remainder after pennies have been issued: " & remainder)
```

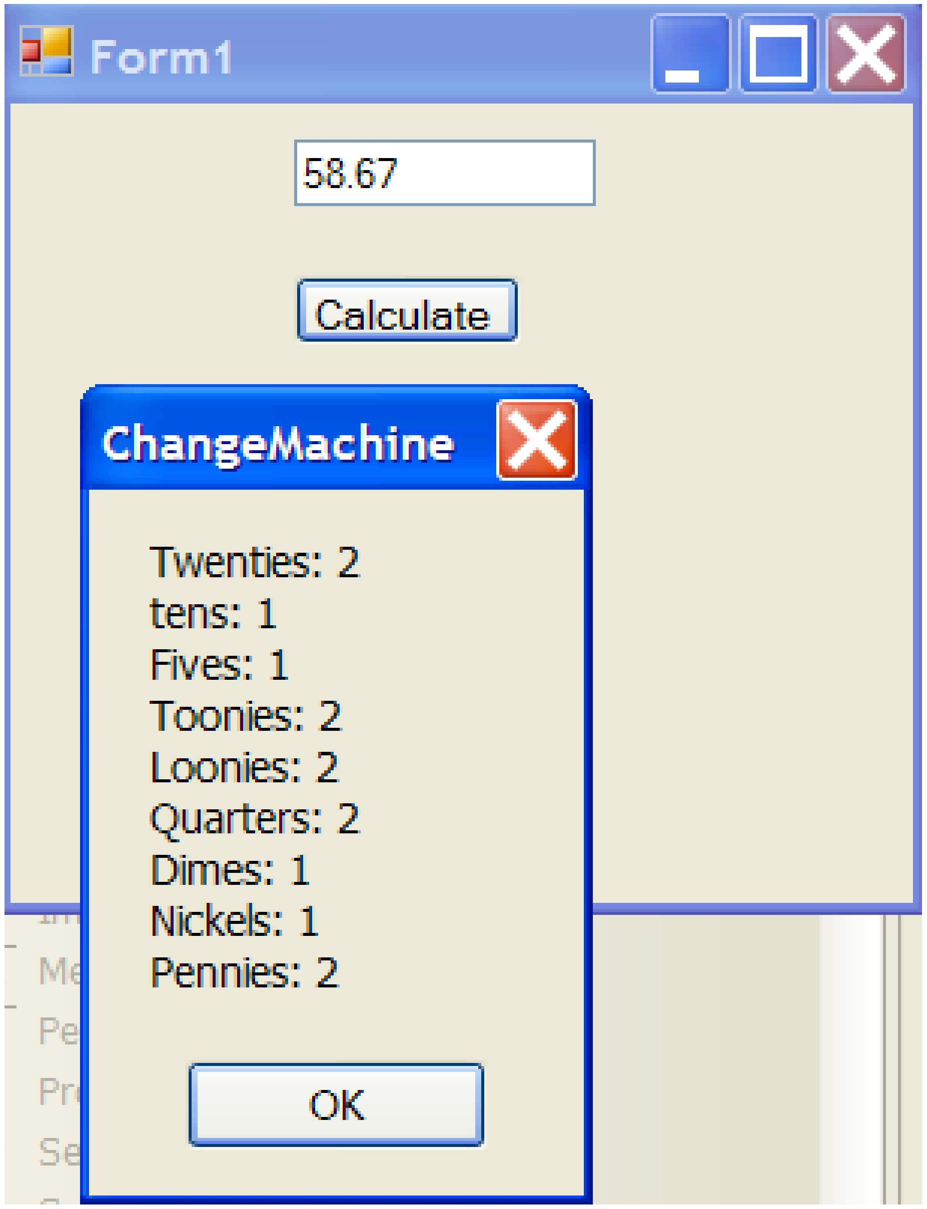
Output

consolidate and concatenate the output



```
Dim output As String
output = "Twenties: " & numTwenties & wrap _
& "tens: " & numTens & wrap _
& "Fives: " & numFives & wrap _
& "Toonies: " & numToonies & wrap _
& "Loonies: " & numLoonies & wrap _
& "Quarters: " & numQuarters & wrap _
& "Dimes: " & numDimes & wrap _
& "Nickels: " & numNickels & wrap _
& "Pennies: " & numPennies

MsgBox(output)
```

Example Output



58.67

```
Twenties: 2
tens: 1
Fives: 1
Toonies: 2
Loonies: 2
Quarters: 2
Dimes: 1
Nickels: 1
Pennies: 2
```