VB Version

**Calculator  
Learn about Reverse Polish Notation by building a simple calculator**

TEACHER’S GUIDE

Created by Richard Pawson

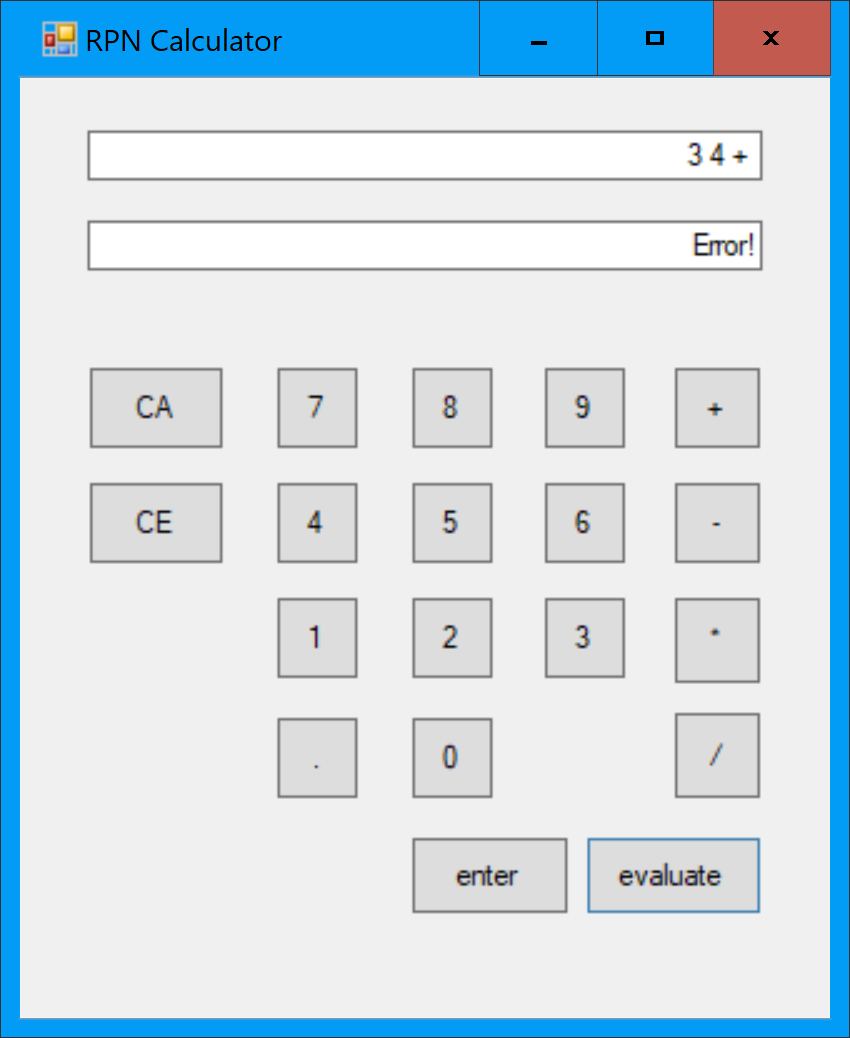
Contents

[Model answers 3](#_Toc508639845)

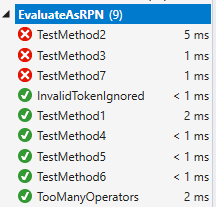
[Complete code for Core.cs 7](#_Toc508639846)

# Model answers

1. Paste in a partial screenshot showing the whole calculator after entering the expression above and hitting evaluate.



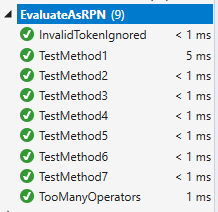
1. Paste in a partial screenshot showing which of the EvaluateAsRPN tests pass, and which fail.



1. Double click on *each test* to examine the test scenario in code. Which operation(s) are causing the failures?

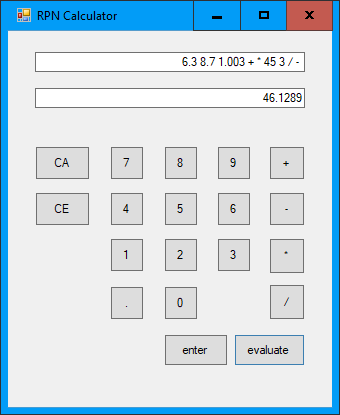
It is failing on the minus and divide operations.

1. Verify that all the EvaluateAsRPN tests pass (the other sets of tests will still fail for now) and paste in a partial screenshot showing this.



1. Paste in a screenshot showing the calculator displaying the full expression and the result.

For example:



1. Show your working by sketching a stack and showing the values on the stack at each step. .

Token processed:

7 6 3 4 + \* -

Stack after each step:

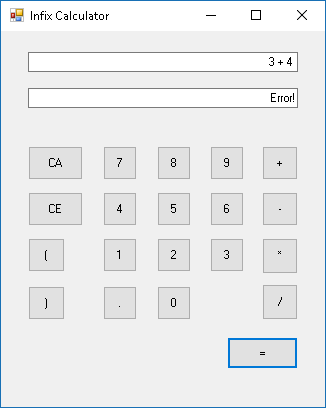
7 6 3 4 7 42 -35 (answer)

7 6 3 6 7

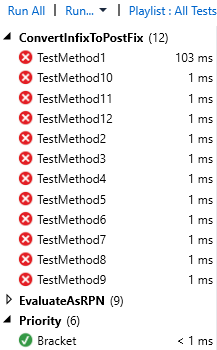
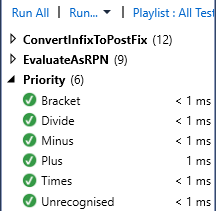
7 6 7

7

1. Paste in a partial screenshot showing the error.



1. Paste in a partial screenshot showing passing and failing tests.



1. Trace though this algorithm on paper using the specific expression shown in the diagram above ( A+B\*C\_D ). What is the order of tokens in the output once the algorithm has been completed?
2. Input: A+B\*C-D

Output:

Siding:

1. Input: +B\*C-D

Output: A

Siding:

1. Input: B\*C-D

Output: A

Siding: +

1. Input: \*C-D

Output: A B

Siding: +

1. Input: C-D

Output: A B

Siding: + \*

1. Input: -D

Output: A B C

Siding: + \*

1. Input: D

Output: A B C \* +

Siding: -

1. Input:

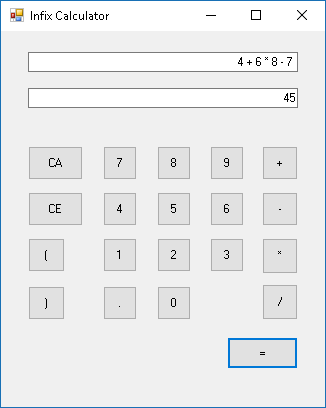
Output: A B C \* + D -

Siding:

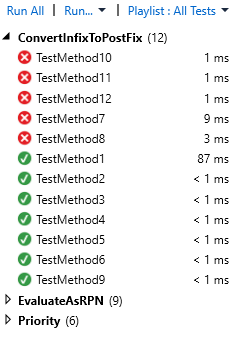
1. In tracing through the algorithm for that same example, what is the maximum number of operators that are being held on the stack at any point?

2

1. Paste in a screenshot showing the expression and the result on the calculator



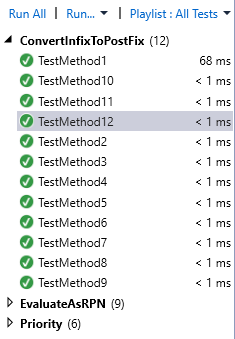
1. Paste in a screenshot showing which tests pass and fail.



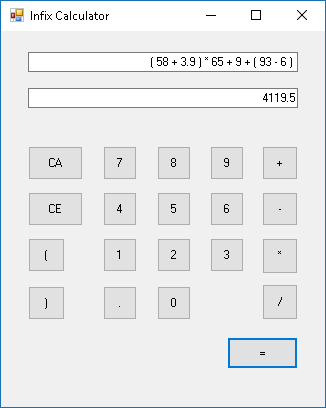
1. Looking at the code for the tests that fail, what do they have in common?

They all contain brackets.

1. Paste in a screenshot showing all tests passing.



1. Paste in a screenshot of the calculator showing both the expression and the result



# Complete code for Core.vb

Imports System.Text

Public Class Core

Private Tokens As List(Of Object) = New List(Of Object)()

Public Sub Clear()

Tokens = New List(Of Object)()

End Sub

Friend Sub AddSymbolAsToken(ByVal symbol As Char)

Tokens.Add(symbol)

End Sub

Public Function AddNumberAsToken(ByVal numberAsText As String) As Double

Dim number As Double = Convert.ToDouble(numberAsText)

Tokens.Add(number)

Return number

End Function

Public Function TokensAsString() As String

Dim sb = New StringBuilder()

For Each token In Tokens

sb.Append(token.ToString()).Append(" ")

Next

Return sb.ToString()

End Function

Public Function EvaluateTokensAsRPN() As Double

Return EvaluateAsRPN(Tokens)

End Function

Public Shared Function EvaluateAsRPN(ByVal Tokens As List(Of Object)) As Double

Dim result As Double = 0

Dim stack = New Stack(Of Double)()

For Each token As Object In Tokens

If TypeOf token Is Double Then

stack.Push(CType(token, Double))

Else 'Assume it Is a Char

Select Case CType(token, Char) 'CType is to ‘cast’ the token into a char

Case "+"c

stack.Push(stack.Pop() + stack.Pop())

Case "-"c

Dim b = stack.Pop()

Dim a = stack.Pop()

stack.Push(a - b)

Case "\*"c

stack.Push(stack.Pop() \* stack.Pop())

Case "/"c

Dim d = stack.Pop()

Dim c = stack.Pop()

stack.Push(c / d)

End Select

End If

Next

result = stack.Pop()

Return result

End Function

Public Function EvaluateTokensAsInfix() As Double

Dim tokensAsRPN = ConvertInfixToPostfix(Tokens)

Return EvaluateAsRPN(tokensAsRPN)

End Function

Public Shared Function ConvertInfixToPostfix(ByVal inputTokens As List(Of Object)) As List(Of Object)

Dim s = New Stack(Of Char)()

Dim outputList = New List(Of Object)()

For Each t In inputTokens

If TypeOf t Is Double Then 'token is a value

outputList.Add(t) 'send it straight to the output

Else 'Is is an operator...

Dim token As Char = CChar(t) '/... so cast it to a char

If token = "("c Then

s.Push(token)

ElseIf token = ")"c Then

Do While s.Count <> 0 AndAlso Not s.Peek().Equals("("c)

outputList.Add(s.Pop())

Loop

s.Pop()

Else

Do While s.Count <> 0 AndAlso Priority(s.Peek()) >= Priority(token)

outputList.Add(s.Pop())

Loop

s.Push(token)

End If

End If

Next t

Do While s.Count <> 0 'Unload any remaining operators onto the stack

outputList.Add(s.Pop())

Loop

Return outputList

End Function

Public Shared Function Priority(ByVal c As Char) As Integer

If c = "\*"c Or c = "/"c Then

Return 2

ElseIf c = "+"c Or c = "-" Then

Return 1

Else

Return 0

End If

End Function

End Class