

# Table of Contents

Abstract.....	2
Introduction .....	3
Method.....	4
Results.....	6
Conclusion .....	9
References.....	10
Appendix .....	11

## **Abstract**

Engineers are problem solvers. The computer sciences enabled interactive learning tools that can be used to assist in our thinking of structural concepts. Here, we propose a Visual Basic program for understanding basic structures such as roofs composed of trusses. Throughout the program, the user may examine one truss at a time and analyze failure related to overloading, and choice of material. The user interface will offer many more possibilities beyond what one can see in a Statics book or from an Engineering Mechanics class. The approach aims to enhance students' visual intuition, and hence understanding of structural analysis and some of the parameters that cause failure of a truss. The program will also allow the user to experiment with different loads, materials, and different truss designs, so that he/she can identify how these three factors correlate with each other. This paper will present some predetermined truss designs used in our program and demonstrate how our application of computer and analytical methods enhance problem solving abilities and contribute to better estimations, resulting in better the structures.

## **Introduction**

Roof failures are at the heart of many insurance claims and it is well known that engineering disasters are often the result of bad estimates. Improper design and overloading are known contributors to the failure of a roof structure, yet they are, with sufficient planning and use of resources, the easiest to prevent. This project describes one of the most convenient methods to prevent failure in the construction of roofs or any structure which consists of trusses. This method is the use of computer programs to make accurate estimates of failure or success of structural members under predetermined conditions. Therefore, we have applied our knowledge in computer techniques to solve the engineering problem responsible for so many failures. This program, the Truss Calculator, is not an in depth analysis of structural failure because truss analysis is a broad and complex subject that requires far more time than we were allowed. This program covers some of the many causes of failure, such as overloading truss members. For most engineering students, it is difficult to visualize what would happen to a structure when the loads applied to it remain constant while the material changes. The Truss Calculator will allow the user to experiment with different loads, materials, and truss designs so that he/she will be able to identify how these three factors correlate with each other. Afterward, it outputs whether the truss will fail or succeed. Also, the graphical interface indicates the failed member in red, so that users will be able to identify which members of the structure failed.

## Method

The method used to calculate the failure or success of each truss under a vertical load is a two-step process. The first step of the algorithm is to calculate the load that each member of the user-defined truss must resist. In the second step of the algorithm, the maximum load for the material chosen is calculated, and compared to each member of the truss to see if any members of the truss are being asked to support more load than the maximum allowed for the chosen material.

Before any calculations are performed the user-input data is scrubbed and calibrated to ensure accurate calculations. They are scrubbed in that checks are performed to make sure that forces are non-negative, lengths and area are greater than zero, and that the factor of safety is a number between 1 and 10. If the data violates these requirements an input window appears prompting the user for new data. When all the data falls within their acceptable ranges, the input is calibrated to SI units of Newtons and meters to prepare them for the algorithms. For the first step of the algorithm, the user is allowed to select one of four common trusses and customize that truss by specifying the height of the truss, the span across the truss, and the vertical loads to which the truss is to be subjected at three different points. Next the forces that each member of the user-defined truss must withstand are calculated. These formulas are calculated individually for each of the four possible truss selections: Howe Bridge, Pratt Bridge, Howe Roof, and Pratt Roof. The algorithms for each truss selection were calculated using the Method of Joints. Each joint of the truss was individually examined and a formula was obtained for each unknown by solving the following equilibrium equations:  $\sum F_x = 0$ ,  $\sum F_y = 0$ . The algorithm for each member of the truss is defined by the user inputs: height, span, and the three vertical forces.

After the loads on all members of the truss are calculated they are stored in an array for comparison with the values calculated in step two.

Step two uses the user input: factor of safety, cross-sectional area, and the type of material chosen to calculate the maximum load that the material can withstand. These algorithms utilize the following relationships:

$$\text{Allowable Stress} = \text{Yield Strength} / \text{Factor of Safety}$$

$$\text{Allowable Load} = \text{Allowable Stress} * \text{Area}$$

Yield Strength is a constant stored by the program for each individual material. The program uses these expressions to calculate the maximum allowable load for the user's selections. Then each member of the truss' loads list is compared to this maximum value. If none of the truss member's loads exceed the maximum allowed value, the truss is a success. If at least one of the truss member's loads exceeds the maximum allowed value the whole truss represents a failure.

The last step of the program is to display the results. The Truss Calculator has a graphical window that draws the selected truss along with the forces that are placed upon it. Below the graphical window the program indicates the result of the truss with a message of "Success" or "Failure". To provide more information to the user, the graphical drawing utilizes color to describe the success and failures of individual truss members. Typically, the truss is drawn in white, however, if a particular member fails, it is drawn in red. This way the user not only knows that the user-defined truss cannot safely resist the forces placed upon it, they also know which members were the cause of failure. This can help the user make modifications to the truss so that it can successfully withstand the desired forces.

## Results – Instructions

Instructions for operating the Truss Calculator:

- 1) Select a Type of Truss
- 2) Select a Material to build the truss
- 3) Enter Dimensions of the Truss: Height, Span
- 4) Enter the Cross-Sectional Area for the members of the truss
- 5) Select Units for Height, Span, and Area
- 6) Enter a Factor of Safety (a number between 1-10, 10 being the safest)
- 7) Enter the forces that you want the truss to withstand
- 8) Select Units for forces
- 9) Select the “Calculate” Button
- 10) Modify input by repeating steps 1-9 until your result is successful and meets the requirements of your project.
- 11) Select “Quit”

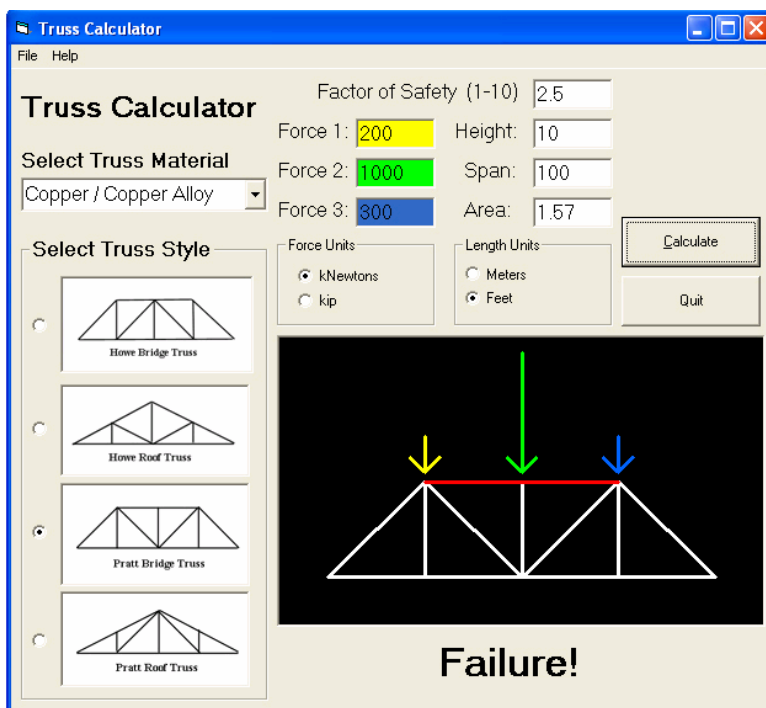


Fig. 1

## Results – Demonstration

As an example of how our program can assist in making truss-related decisions, and to see how the material, form, and design determine whether truss will fail, suppose you were given the following

specifications: A Bridge Truss that can span 100 ft, be no taller than 10 ft, that can withstand forces of 200 kN on the left side, 1000 kN in the middle, and 300 kN on the right, to within a factor of safety of 2.5. The budget is limiting on this project and you can only afford a copper alloy material with a cross-sectional area of 1.57 ft<sup>2</sup>. Selecting the Pratt Bridge and entering this data into the Truss calculator, we can see that the selected truss fails under the required load.

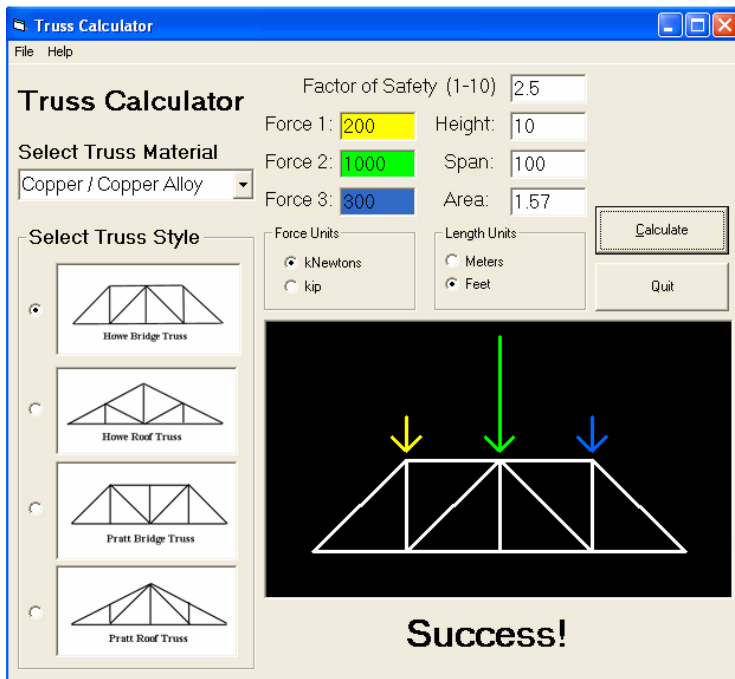


Fig. 2

(Fig.1):

With no budget to make improvements what can we do? By selecting a new truss, the Howe Bridge, we can see that the Howe Bridge is better designed to handle the large central load without changing any specifications (Fig. 2).

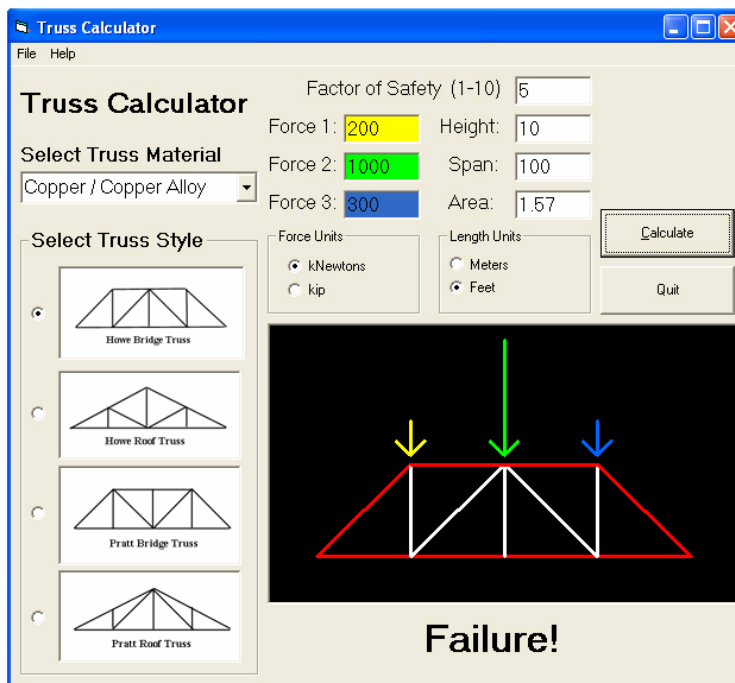


Fig. 3

Now let's say that the client has increased their demands on the requirement of safety. The bridge must now satisfy and Factor of Safety of 5. Our current specifications fail under these new requirements (Fig.3). Notice that none of the actual loads have

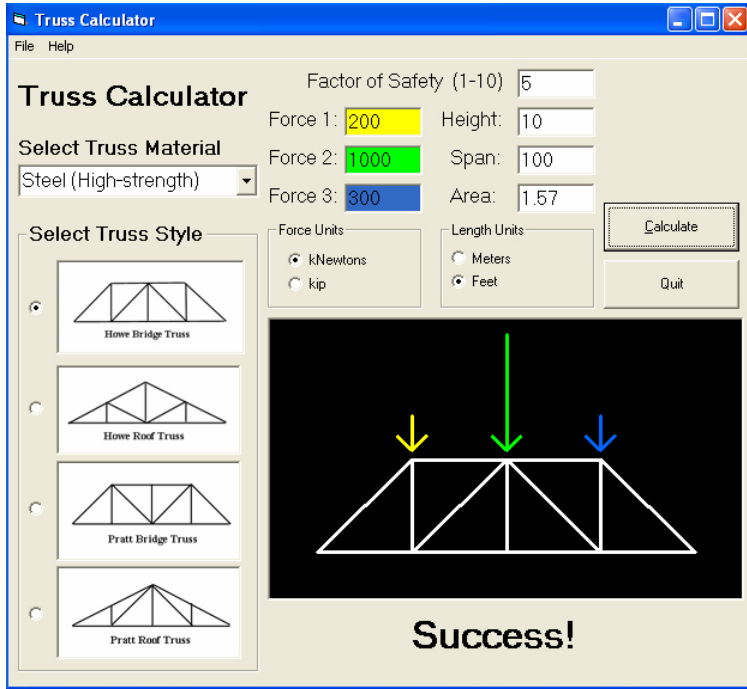


Fig. 4

changed –only the range of what is considered an acceptable loads has changed. Luckily, this increase in demand comes with an increase of budget. Therefore, instead of using copper-alloy, we opt to purchase a stronger material: High-Strength Steel, resulting in “Success!”(Fig. 4)

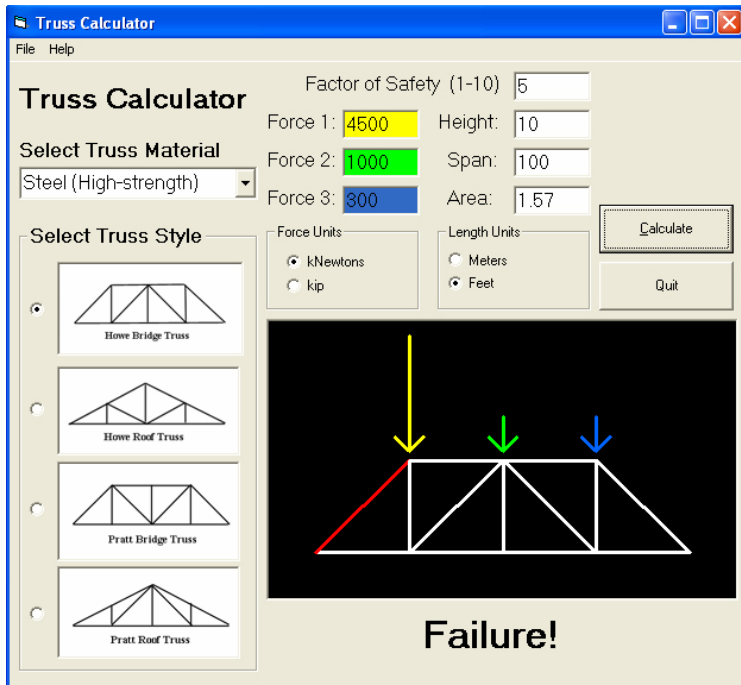


Fig. 5

Lastly the engineers have changed the function of the truss so that it now must accommodate a load of 4500 kN on the left side. Again our present design fails to meet the new demands (Fig.5):

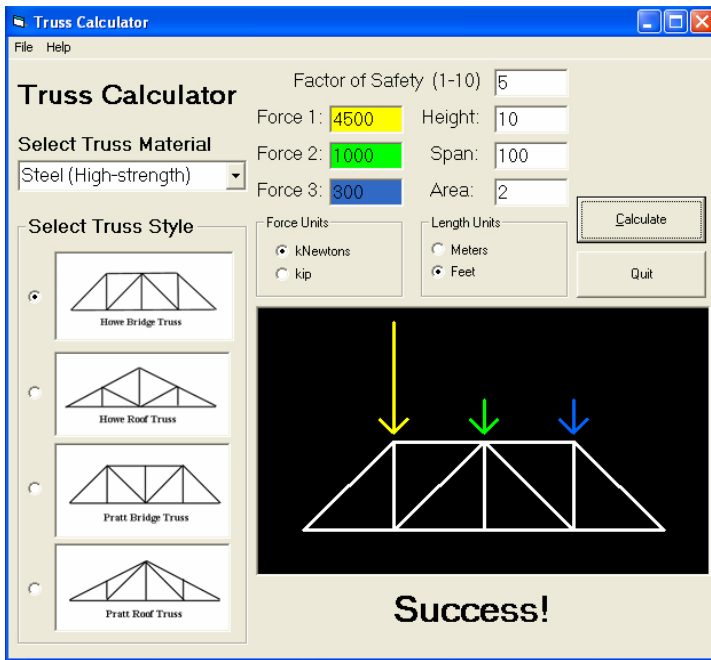


Fig. 6

Supposing that there is a little money left in our budget increase, we imagine that we can afford to use steel members with a slightly larger cross-sectional area. To see if this will solve the problem, the area is changed from 1.57 ft<sup>2</sup> to 2 ft<sup>2</sup>. (Fig. 6)

Now once more, the truss has been redesigned to meet all the revised specifications with instant results and eliminating tedious and repetitive calculations.

## Conclusion

Throughout the making of this program, we learned how to generalize a solution to make the load calculations work with any user input of materials, forces, and dimensions. Computers methods are for solving complex problems like determining whether or not a truss will fail. In order to output “failure” or “success” for a given truss, the computer code performs a series of calculations. Furthermore, this program is highly effective because of its efficiency and user friendly interface. This program allows the user to select one of four truss styles, to select from an archive of materials, and to apply their choice of external forces to three different points. Then, the program will not only determine whether the truss fails or not, but will also show the user exactly which member is at fault. This program is effective because solving truss systems is time consuming without a computer. Solving numerous amounts of varying truss systems will become tedious fast.

Not only is the program able to solve the systems immediately, it also allows for large amounts of variability with a number of possible combinations to create. The user has the power to change different variables, all of which have a factor in whether the truss will fail or not. Even though this program seems very complex, there is still more that can be added to make this program more valuable and effective. To simplify the algorithm, the only external forces that we applied were vertical forces applied to the top. Therefore some improvements could be to add the possibility of applying horizontal forces for a more realistic simulation. Another suggestion is to be able to change the position of the applied forces from top to bottom or from left to right, since bridge trusses are ordinarily loaded from the bottom. Given that structural analysis is a complicated subject, we assumed that all trusses represented in the program stand on a roller support on one side, and a pin support on the other side. One way to add more variety to the program is by having a choice of different support types. And, another way to enhance the program is the addition of a larger selection of trusses and materials. All possible strategies should be used to reduce the risk of failure to a tolerable level at an acceptable cost. This program is one such strategy. As engineering students, we realized that this project helped us to better approach engineering problems and to apply our knowledge of computer and analytical methods.

## References

- Bedford, Anthony and Wallace Fowler, Statics: Engineering Mechanics 4 ed., Prentice Hall; Upper Saddle River, New Jersey, 2005.
- James M. Gere Mechanics of Materials 6ed. Thomson-Engineering, Belmont, CA, 2003.
- Mitchell C. Kerman and Ronald L. Computer programming fundamentals with applications in Brown, Visual Basic 6.0. Addison Wesley, New York, 1999

## Index: Program Code

Option Explicit

Const NIL = -1

Const HoweBridge = 0

Const HoweRoof = 1

Const PrattBridge = 2

Const PrattRoof = 3

Const ftToMeter = 0.3048 'm/ft

Const lbToNewton = 4.448 'N/lb

Const kipToNewton = 4448 'N/kip

Const MPasToPas = 1000000# 'Pas/MPas

Const kNtoN = 1000# 'N/kN

'Yield Stress (MPa) (lowest value)

Const Al2014\_T6Yield = 410

Const Al6061\_T6Yield = 270

Const Al7075\_T6Yield = 480

Const BrassYield = 70

Const BronzeYield = 82

Const CuYield = 55

Const MgYield = 80

Const MonelYield = 170

Const NiYield = 100

Const SteelHSYield = 340

Const SteelMachineYield = 340

Const SteelSpringYield = 400

Const SteelStainlessYield = 280

Const SteelToolYield = 520

Const SteelASTM\_A36Yield = 250

Const SteelASTM\_A572Yield = 340

Const SteelASTM\_A514Yield = 700

Const TiYield = 760

Const WoodDouglasYield = 30

Const WoodOakYield = 30

Const WoodSPineYield = 30

Dim trussSelection As Integer

Dim factorSafety As Double

Dim f1 As Double, f2 As Double, f3 As Double

Dim area As Double

Dim trussHeight As Double, span As Double

Dim allowableLoad As Double

Dim trussForces(1 To 13) As Double

Private Type joint

x As Double

y As Double

dir(1 To 5) As Integer

```
members(1 To 5) As Integer
End Type
```

```
Private Sub drawTruss(truss() As joint)
'Generates graphics for display window
```

```
Dim i As Integer, j As Integer
Dim joist As Integer, index As Integer
Dim R As Integer, G As Integer, B As Integer
```

```
picOutput.Scale (-1, -5)-(9, 1)
picOutput.DrawWidth = 3
```

```
picOutput.Cls
'Draw the truss
For i = 1 To 8
  For j = 1 To 5
```

```
    joist = truss(i).dir(j)
    If (joist <> NIL) Then
      index = truss(i).members(j)
      If (Abs(trussForces(index)) > allowableLoad) Then
        'use red if failure
        R = 255
        B = 0
        G = 0
      Else
        'Else use white
        R = 255
        B = 255
        G = 255
      End If
      picOutput.Line (truss(i).x, -truss(i).y)-(truss(joist).x, -truss(joist).y), RGB(R, G, B)
    End If
```

```
  Next j
Next i
```

```
'Draw the force arrows
Dim point As Double, tail As Double
Dim length As Double
Dim arrowScale As Double
```

```
If (f1 > f2) And (f1 > f3) Then
  arrowScale = f1 / 2.5
Else
  If (f2 > f3) Then
    arrowScale = f2 / 2.5
  Else
    arrowScale = f3 / 2.5
  End If
End If
```

```
point = -2.2
If (f2 <> 0) Then
  length = f2 / arrowScale
```

```

If (length < 0.75) Then length = 0.75
If (length > 2.5) Then length = 2.5
tail = point - length
picOutput.Line (4, point)-(4, tail), RGB(0, 255, 0)
picOutput.Line (3.7, point - 0.3)-(4, point), RGB(0, 255, 0)
picOutput.Line (4, point)-(4.3, point - 0.3), RGB(0, 255, 0)
End If

```

```

If (trussSelection = HoweRoof) Or (trussSelection = PrattRoof) Then point = -1.2

```

```

If (f1 <> 0) Then
    length = f1 / arrowScale
    If (length < 0.75) Then length = 0.75
    If (length > 2.5) Then length = 2.5
    tail = point - length
    picOutput.Line (2, point)-(2, tail), RGB(255, 255, 0)
    picOutput.Line (1.7, point - 0.3)-(2, point), RGB(255, 255, 0)
    picOutput.Line (2, point)-(2.3, point - 0.3), RGB(255, 255, 0)
End If

```

```

If (f3 <> 0) Then
    length = f3 / arrowScale
    If (length < 0.75) Then length = 0.75
    If (length > 2.5) Then length = 2.5
    tail = point - length
    picOutput.Line (6, point)-(6, tail), RGB(0, 100, 255)
    picOutput.Line (5.7, point - 0.3)-(6, point), RGB(0, 100, 255)
    picOutput.Line (6, point)-(6.3, point - 0.3), RGB(0, 100, 255)
End If

```

```

End Sub

```

```

Private Sub initializeTruss(truss() As joint)

```

```

'   H   G   F
'   -----
'  /|   |   |\
' /|   |   |\
' /|   |   |\
'/_|_|_|_|_|_\
'A  B  C  D  E

```

```

Dim i As Integer
Dim j As Integer

```

```

Select Case trussSelection
    Case HoweBridge
        Open "C:/TrussCalc/HoweBridge.dat" For Input As #3
    Case HoweRoof
        Open "C:/TrussCalc/HoweRoof.dat" For Input As #3
    Case Is = PrattBridge
        Open "C:/TrussCalc/PrattBridge.dat" For Input As #3
    Case Is = PrattRoof
        Open "C:/TrussCalc/PrattRoof.dat" For Input As #3
End Select

```

```

'Load the drawing coordinates

```

```

For i = 1 To 8
    Input #3, truss(i).x
    Input #3, truss(i).y
Next i

'Load directions to connecting joints
For i = 1 To 8
    For j = 1 To 5
        Input #3, truss(i).dir(j)
    Next j
Next i

'Load truss members to correspond with list of trussForce
For i = 1 To 8
    For j = 1 To 5
        Input #3, truss(i).members(j)
    Next j
Next i

Close #3

End Sub

Sub isGoodData()
'Ensures that all user-input meets the needs of the program _
and initializes data for algorithms

If (optHoweBridge) Then trussSelection = HoweBridge
If (optHoweRoof) Then trussSelection = HoweRoof
If (optPrattBridge) Then trussSelection = PrattBridge
If (optPrattRoof) Then trussSelection = PrattRoof

factorSafety = CDbI(Val(txtSafety.Text))
Do While (factorSafety < 1) Or (10 < factorSafety)
    factorSafety = CDbI(Val(InputBox("Safety Factor must be a value from 1 to 10.")))
    txtSafety.Text = factorSafety
Loop

f1 = CDbI(Val(txtF1.Text))
Do While (f1 < 0)
    f1 = CDbI(Val(InputBox("Force #1 must be a positive value.")))
Loop
txtF1.Text = f1

f2 = CDbI(Val(txtF2.Text))
Do While (f2 < 0)
    f2 = CDbI(Val(InputBox("Force #2 must be a positive value.")))
Loop
txtF2.Text = f2

f3 = CDbI(Val(txtF3.Text))
Do While (f3 < 0)
    f3 = CDbI(Val(InputBox("Force #3 must be a positive value.")))
Loop
txtF3.Text = f3

```

```

trussHeight = CDbI(Val(txtHeight.Text))
Do While (trussHeight <= 0)
    trussHeight = CDbI(Val(InputBox("Height must be a positive, non-zero value.")))
Loop
txtHeight.Text = trussHeight

```

```

span = CDbI(Val(txtSpan.Text))
Do While (span <= 0)
    span = CDbI(Val(InputBox("Span must be a positive, non-zero value.")))
Loop
txtSpan.Text = span

```

```

area = CDbI(Val(txtArea.Text))
Do While (area <= 0)
    area = CDbI(Val(InputBox("Area must be a positive, non-zero value.")))
Loop
txtArea.Text = area

```

```

'Convert all input to SI units: (N), (m)
If (optkNewtons) Then
    f1 = f1 * kNtoN
    f2 = f2 * kNtoN
    f3 = f3 * kNtoN
End If

```

```

If (optKIP) Then
    f1 = f1 * kipToNewton
    f2 = f2 * kipToNewton
    f3 = f3 * kipToNewton
End If

```

```

If (optFt) Then
    trussHeight = trussHeight * ftToMeter
    span = span * ftToMeter
    area = area * ftToMeter ^ 2
End If

```

```
End Sub
```

```
Sub calculateTrussForces()
'Calculates the load placed on each individual member of the truss

```

```

Dim cosTheta As Double, sinTheta As Double
Dim cosAlpha As Double, sinAlpha As Double
Dim hypotenuse As Double
Dim n1 As Double, n2 As Double

```

```

Dim AB As Double, AH As Double, BC As Double, BG As Double, BH As Double, _
CD As Double, CF As Double, CG As Double, CH As Double, DE As Double, DF As Double, DG As
Double, _
EF As Double, FG As Double, GH As Double

```

```

Select Case trussSelection
Case HoweBridge
    hypotenuse = Sqr(trussHeight ^ 2 + (span / 4) ^ 2)

```

$\cos\theta = \text{span} / (4 * \text{hypotenuse})$   
 $\sin\theta = \text{trussHeight} / \text{hypotenuse}$   
 $n2 = f1 / 4 + f2 / 2 + 3 * f3 / 4$   
 $n1 = f1 + f2 + f3 - n2$   
 $AH = -n1 / \sin\theta$   
 $AB = -AH * \cos\theta$   
 $GH = AH * \cos\theta$   
 $BH = -AH * \sin\theta - f1$   
 $BG = -BH / \sin\theta$   
 $BC = AB - BG * \cos\theta$   
 $CG = 0$   
 $EF = -n2 / \sin\theta$   
 $DE = -EF * \cos\theta$   
 $FG = EF * \cos\theta$   
 $DF = -EF * \sin\theta - f3$   
 $DG = -DF / \sin\theta$   
 $CD = DE - DG * \cos\theta$   
 $\text{trussForces}(1) = AB$   
 $\text{trussForces}(2) = AH$   
 $\text{trussForces}(3) = BC$   
 $\text{trussForces}(4) = BG$   
 $\text{trussForces}(5) = BH$   
 $\text{trussForces}(6) = CD$   
 $\text{trussForces}(7) = CG$   
 $\text{trussForces}(8) = DE$   
 $\text{trussForces}(9) = DF$   
 $\text{trussForces}(10) = DG$   
 $\text{trussForces}(11) = EF$   
 $\text{trussForces}(12) = FG$   
 $\text{trussForces}(13) = GH$

#### Case HoweRoof

$\text{hypotenuse} = \text{Sqr}((\text{trussHeight} / 2)^2 + (\text{span} / 4)^2)$   
 $\cos\theta = \text{span} / (4 * \text{hypotenuse})$   
 $\sin\theta = \text{trussHeight} / (2 * \text{hypotenuse})$   
 $n2 = f1 / 4 + f2 / 2 + 3 * f3 / 4$   
 $n1 = f1 + f2 + f3 - n2$   
 $AH = -n1 / \sin\theta$   
 $AB = -AH * \cos\theta$   
 $BH = 0$   
 $BC = AB$   
 $CH = (-f1 - BH) / (2 * \sin\theta)$   
 $GH = AH - CH$   
 $CG = -f2$   
 $EF = -n2 / \sin\theta$   
 $DE = -EF * \cos\theta$   
 $DF = 0$   
 $CD = DE$   
 $CF = (-f3 - DF) / (2 * \sin\theta)$   
 $FG = EF - CF$   
 $\text{trussForces}(1) = AB$   
 $\text{trussForces}(2) = AH$   
 $\text{trussForces}(3) = BC$   
 $\text{trussForces}(4) = BH$   
 $\text{trussForces}(5) = CD$   
 $\text{trussForces}(6) = CF$   
 $\text{trussForces}(7) = CG$

trussForces(8) = CH  
 trussForces(9) = DE  
 trussForces(10) = DF  
 trussForces(11) = EF  
 trussForces(12) = FG  
 trussForces(13) = GH

#### Case PrattBridge

hypotenuse =  $\text{Sqr}(\text{trussHeight}^2 + (\text{span} / 4)^2)$   
 cosTheta =  $\text{span} / (4 * \text{hypotenuse})$   
 sinTheta =  $\text{trussHeight} / \text{hypotenuse}$   
 $n2 = f1 / 4 + f2 / 2 + 3 * f3 / 4$   
 $n1 = f1 + f2 + f3 - n2$   
 AH =  $-n1 / \text{sinTheta}$   
 AB =  $-AH * \text{cosTheta}$   
 BC = AB  
 BH = 0  
 CH =  $(-f1 - BH) / \text{sinTheta} - AH$   
 GH =  $(AH - CH) / \text{cosTheta}$   
 CG =  $-f2$   
 EF =  $-n2 / \text{sinTheta}$   
 DE =  $-EF * \text{cosTheta}$   
 CD = DE  
 DF = 0  
 CF =  $(-f3 - DF) / \text{sinTheta} - EF$   
 FG =  $(EF - CF) / \text{cosTheta}$   
 trussForces(1) = AB  
 trussForces(2) = AH  
 trussForces(3) = BC  
 trussForces(4) = BH  
 trussForces(5) = CD  
 trussForces(6) = CF  
 trussForces(7) = CG  
 trussForces(8) = CH  
 trussForces(9) = DE  
 trussForces(10) = DF  
 trussForces(11) = EF  
 trussForces(12) = FG  
 trussForces(13) = GH

#### Case PrattRoof

hypotenuse =  $\text{Sqr}(\text{trussHeight}^2 + (\text{span} / 2)^2)$   
 cosAlpha =  $\text{span} / (2 * \text{hypotenuse})$   
 sinAlpha =  $\text{trussHeight} / \text{hypotenuse}$   
 hypotenuse =  $\text{Sqr}(\text{trussHeight}^2 + (\text{span} / 4)^2)$   
 cosTheta =  $\text{span} / (4 * \text{hypotenuse})$   
 sinTheta =  $\text{trussHeight} / \text{hypotenuse}$   
 $n2 = f1 / 4 + f2 / 2 + 3 * f3 / 4$   
 $n1 = f1 + f2 + f3 - n1$   
 AH =  $-n1 / \text{sinAlpha}$   
 AB =  $-AH * \text{cosAlpha}$   
 GH = AH  
 BH =  $-AH * \text{sinAlpha} + GH * \text{sinAlpha} + f3$   
 BG =  $-BH / \text{sinTheta}$   
 BC = AB - BG  
 CG = 0  
 CD = BC  
 EF =  $-n2 / \text{sinAlpha}$

```

DE = -EF * cosAlpha
FG = EF
DG = (DE - CD) / cosTheta
DF = -DG / sinTheta
trussForces(1) = AB
trussForces(2) = AH
trussForces(3) = BC
trussForces(4) = BG
trussForces(5) = BH
trussForces(6) = CD
trussForces(7) = CG
trussForces(8) = DE
trussForces(9) = DF
trussForces(10) = DG
trussForces(11) = EF
trussForces(12) = FG
trussForces(13) = GH
End Select

```

End Sub

Function Fail(trussForces() As Double) As Boolean  
'Returns TRUE if the truss fails to meet the safety requirements, \_  
otherwise returns FALSE

```

Dim i As Integer
Dim yieldStrength As Double
Dim allowableStress As Double

```

```

Select Case IstMaterial
Case "Aluminum Alloy 2014-T6"
    yieldStrength = Al2014_T6Yield
Case "Aluminum Alloy 6061-T6"
    yieldStrength = Al6061_T6Yield
Case "Aluminum Alloy 7075-T6"
    yieldStrength = Al7075_T6Yield
Case "Brass"
    yieldStrength = BrassYield
Case "Bronze"
    yieldStrength = BronzeYield
Case "Copper / Copper Alloy"
    yieldStrength = CuYield
Case "Magnesium Alloy"
    yieldStrength = MgYield
Case "Monel (67% Ni, 30% Cu)"
    yieldStrength = MonelYield
Case "Nickel"
    yieldStrength = NiYield
Case "Steel (High-strength)"
    yieldStrength = SteelHSYield
Case "Steel (Machine)"
    yieldStrength = SteelMachineYield
Case "Steel (Spring)"
    yieldStrength = SteelSpringYield
Case "Steel (Stainless)"
    yieldStrength = SteelStainlessYield

```

```

Case "Steel (Tool)"
    yieldStrength = SteelToolYield
Case "Steel ASTM-A36"
    yieldStrength = SteelASTM_A36Yield
Case "Steel ASTM-A572"
    yieldStrength = SteelASTM_A572Yield
Case "Steel ASTM-A514"
    yieldStrength = SteelASTM_A514Yield
Case "Titanium Alloy"
    yieldStrength = TiYield
Case "Wood (Douglas fir)"
    yieldStrength = WoodDouglasYield
Case "Wood (Oak)"
    yieldStrength = WoodOakYield
Case "Wood (Southern pine)"
    yieldStrength = WoodSPineYield
Case Else
    MsgBox "Error in YieldStrength assignment!", vbExclamation, "ERROR"
End Select

'from materials text-book
yieldStrength = yieldStrength * MPasToPas    'pascals
allowableStess = yieldStrength / factorSafety 'pascals
allowableLoad = allowableStess * area        'N = pascals / m^2

'temporary test
' picTest.Print allowableLoad

'Compare Truss Forces against the material's allowable load to determine if the truss will fail
Fail = False
For i = 1 To 13
    If (Abs(trussForces(i)) > allowableLoad) Then Fail = True
Next i

End Function

Private Sub cmdCalculate_Click()

    Dim truss(1 To 8) As joint

    Call isGoodData

    Call initializeTruss(truss())
    Call calculateTrussForces

    If Not Fail(trussForces()) Then
        lblResult.Caption = "Success!"
    Else
        lblResult.Caption = "Failure!"
    End If

' picTest.Cls
' Dim i As Integer
' For i = 1 To 13
'     picTest.Print i & ") " & trussForces(i)
' Next i

```

```
' picTest.Print "Fail: "; Fail(trussForces())  
  Call drawTruss(truss())
```

```
End Sub
```

```
Private Sub mnuAbout_Click()
```

```
  MsgBox "Truss Calculator (version 1.0); Programmers: (Structure 1 Team) Mark Edward Daley, Carolina  
  J. Rodriguez, Raul Barajas", vbOKOnly, "ABOUT TRUSS CALCULATOR"
```

```
End Sub
```

```
Private Sub mnuOpen_Click()
```

```
  Close  
  CommonDialog1.ShowOpen  
  Open CommonDialog1.FileName For Input As #1
```

```
End Sub
```

```
Private Sub mnuSave_Click()
```

```
  CommonDialog1.ShowSave  
  Open CommonDialog1.FileName For Output As #2
```

```
  Close #2
```

```
End Sub
```

```
Private Sub mnuExit_Click()
```

```
  Close  
  End
```

```
End Sub
```

```
Private Sub cmdQuit_Click()
```

```
  Close  
  End
```

```
End Sub
```

```
Private Sub Form_Load()
```

```
  optHoweBridge = True  
  optkNewtons = True
```

```
End Sub
```

