Program Cover Sheet

|  |
| --- |
| Name: Austin Rippee |
| Assignment: Assignment 9 |
| List any parts of the assignment that do not work/were not completed:  Everything seems to be working smoothly. If there are things missing/wrong, then it has gone over my head. |

|  |
| --- |
| Instructor’s Comments: |
| Grade: |

Program Submission Requirements: (1) all files, zipped and uploaded to Canvas and (2) a completed cover sheet, program execution screenshots and source code printed, **stapled** and turned in during class. Failure to follow the submission requirements will result in points lost on that particular assignment.

Imports Microsoft.Office.Interop

'------------------------------------------------------------

'- File Name : StudentGrades.vb -

'- Part of Project: Main -

'------------------------------------------------------------

'- Written By: Austin Rippee -

'- Written On: April 10, 2022 -

'------------------------------------------------------------

'- File Purpose: -

'- The user will be prompted with a console app which opens

'- an excel application and displays the data

'------------------------------------------------------------

'- Program Purpose: -

'- -

'- This program displays students initials, last name, and grades

'- in an excel file

'------------------------------------------------------------

'- Global Variable Dictionary (alphabetically): -

'- excelApp - Excel application instance

'------------------------------------------------------------

Module StudentGrades

'Creates the excel application instance

Dim excelApp As Excel.Application

Public Class clsStudent

Private strInitials As String = ""

Private strlastName As String = ""

Private sngScores(3) As Single

Private dblExamScore As Double

'------------------------------------------------------------

'- Subprogram Name: New() -

'------------------------------------------------------------

'- Written By: Austin Rippee

'- Written On: April 10, 2022 -

'------------------------------------------------------------

'- Subprogram Purpose: -

'- -

'- Creates the default constructor for clsStudent

'------------------------------------------------------------

'- Parameter Dictionary (in parameter order): -

'- (None)

'------------------------------------------------------------

'- Local Variable Dictionary (alphabetically): -

'- (None) -

'------------------------------------------------------------

Public Sub New()

setInitials("")

setLastName("")

setScores({0, 0, 0, 0})

setExamScore(0.0)

End Sub

'------------------------------------------------------------

'- Subprogram Name: New() -

'------------------------------------------------------------

'- Written By: Austin Rippee

'- Written On: April 10, 2022 -

'------------------------------------------------------------

'- Subprogram Purpose: -

'- -

'- Creates the named constructor for clsStudent

'------------------------------------------------------------

'- Parameter Dictionary (in parameter order): -

'- newExamScore - Double to keep track of the student's exam score

'- newInitials - string for the student's initials

'- newLastName - string for the student's last name

'- newScores() - array of singles to keep track of the scores

'------------------------------------------------------------

'- Local Variable Dictionary (alphabetically): -

'- (None) -

'------------------------------------------------------------

Public Sub New(ByVal newInitials As String, ByVal newLastName As String, ByVal newScores() As Single, ByVal newExamScore As Double)

setInitials(newInitials)

setLastName(newLastName)

setScores(newScores)

setExamScore(newExamScore)

End Sub

'------------------------------------------------------------

'- Subprogram Name: setInitials -

'------------------------------------------------------------

'- Written By: Austin Rippee

'- Written On: April 10, 2022 -

'------------------------------------------------------------

'- Subprogram Purpose: -

'- -

'- Creates the setter for initials

'------------------------------------------------------------

'- Parameter Dictionary (in parameter order): -

'- newInitials - initials of student

'------------------------------------------------------------

'- Local Variable Dictionary (alphabetically): -

'- (None) -

'------------------------------------------------------------

Public Sub setInitials(ByVal newInitials As String)

strInitials = newInitials

End Sub

'------------------------------------------------------------

'- Function Name: getInitials -

'------------------------------------------------------------

'- Written By: Austin Rippee

'- Written On: April 10, 2022 -

'------------------------------------------------------------

'- Subprogram Purpose: -

'- -

'- Creates the getter for initials

'------------------------------------------------------------

'- Parameter Dictionary (in parameter order): -

'- (None)

'------------------------------------------------------------

'- Local Variable Dictionary (alphabetically): -

'- (None) -

'------------------------------------------------------------

'- Returns: -

'- String – the initials of the student -

'------------------------------------------------------------

Public Function getInitials() As String

Return strInitials

End Function

'------------------------------------------------------------

'- Subprogram Name: setLastName -

'------------------------------------------------------------

'- Written By: Austin Rippee

'- Written On: April 10, 2022 -

'------------------------------------------------------------

'- Subprogram Purpose: -

'- -

'- Creates the setter for last name

'------------------------------------------------------------

'- Parameter Dictionary (in parameter order): -

'- newLastName - last name of student

'------------------------------------------------------------

'- Local Variable Dictionary (alphabetically): -

'- (None) -

'------------------------------------------------------------

Public Sub setLastName(ByVal newLastName As String)

strlastName = newLastName

End Sub

'------------------------------------------------------------

'- Function Name: getLastName -

'------------------------------------------------------------

'- Written By: Austin Rippee

'- Written On: April 10, 2022 -

'------------------------------------------------------------

'- Subprogram Purpose: -

'- -

'- Creates the getter for last name

'------------------------------------------------------------

'- Parameter Dictionary (in parameter order): -

'- (None)

'------------------------------------------------------------

'- Local Variable Dictionary (alphabetically): -

'- (None) -

'------------------------------------------------------------

'- Returns: -

'- String – the last name of the student -

'------------------------------------------------------------

Public Function getLastName() As String

Return strlastName

End Function

'------------------------------------------------------------

'- Subprogram Name: setScores -

'------------------------------------------------------------

'- Written By: Austin Rippee

'- Written On: April 10, 2022 -

'------------------------------------------------------------

'- Subprogram Purpose: -

'- -

'- Creates the setter for student scores

'------------------------------------------------------------

'- Parameter Dictionary (in parameter order): -

'- newScores - scores of student

'------------------------------------------------------------

'- Local Variable Dictionary (alphabetically): -

'- (None) -

'------------------------------------------------------------

Public Sub setScores(ByVal newScores As Single())

sngScores = newScores

End Sub

'------------------------------------------------------------

'- Function Name: getScores -

'------------------------------------------------------------

'- Written By: Austin Rippee

'- Written On: April 10, 2022 -

'------------------------------------------------------------

'- Subprogram Purpose: -

'- -

'- Creates the getter for student scores

'------------------------------------------------------------

'- Parameter Dictionary (in parameter order): -

'- (None)

'------------------------------------------------------------

'- Local Variable Dictionary (alphabetically): -

'- (None) -

'------------------------------------------------------------

'- Returns: -

'- Single – Array of singles to return the scores -

'------------------------------------------------------------

Public Function getScores() As Single()

Return sngScores

End Function

'------------------------------------------------------------

'- Subprogram Name: setExamScore -

'------------------------------------------------------------

'- Written By: Austin Rippee

'- Written On: April 10, 2022 -

'------------------------------------------------------------

'- Subprogram Purpose: -

'- -

'- Creates the setter for student exam score

'------------------------------------------------------------

'- Parameter Dictionary (in parameter order): -

'- newExamScore - last name of student

'------------------------------------------------------------

'- Local Variable Dictionary (alphabetically): -

'- (None) -

'------------------------------------------------------------

Public Sub setExamScore(ByVal newExamScore As String)

dblExamScore = newExamScore

End Sub

'------------------------------------------------------------

'- Function Name: getExamScore -

'------------------------------------------------------------

'- Written By: Austin Rippee

'- Written On: April 10, 2022 -

'------------------------------------------------------------

'- Subprogram Purpose: -

'- -

'- Creates the getter for exam score

'------------------------------------------------------------

'- Parameter Dictionary (in parameter order): -

'- (None)

'------------------------------------------------------------

'- Local Variable Dictionary (alphabetically): -

'- (None) -

'------------------------------------------------------------

'- Returns: -

'- Double – the exam score of the student -

'------------------------------------------------------------

Public Function getExamScore() As Double

Return dblExamScore

End Function

'------------------------------------------------------------

'- Subprogram Name: Add -

'------------------------------------------------------------

'- Written By: Austin Rippee

'- Written On: April 10, 2022 -

'------------------------------------------------------------

'- Subprogram Purpose: -

'- -

'- Adds a new instance of clsStudent

'------------------------------------------------------------

'- Parameter Dictionary (in parameter order): -

'- myStudents - instance of clsStudent

'------------------------------------------------------------

'- Local Variable Dictionary (alphabetically): -

'- (None) -

'------------------------------------------------------------

Public Sub Add(clsStudent As clsStudent)

'Adds a new instance

Dim myStudents As New clsStudent("", "", {0, 0, 0, 0}, 0.0)

End Sub

End Class

'------------------------------------------------------------

'- Subprogram Name: sDisplayStudents -

'------------------------------------------------------------

'- Written By: Austin Rippee

'- Written On: April 10, 2022 -

'------------------------------------------------------------

'- Subprogram Purpose: -

'- -

'- Displays the string text of the student data

'------------------------------------------------------------

'- Parameter Dictionary (in parameter order): -

'- studentCol - total number of students

'------------------------------------------------------------

'- Local Variable Dictionary (alphabetically): -

'- (None) -

'------------------------------------------------------------

Public Sub DisplayStudents(ByVal studentCol As IEnumerable(Of clsStudent))

'Displays the list of students by checking each Ienumerable

For Each st As clsStudent In studentCol

Console.WriteLine(st.getInitials & vbTab & st.getLastName & vbTab & st.getScores(0) & vbTab & st.getScores(1) & vbTab & st.getScores(2) & vbTab & st.getScores(3) & vbTab & st.getExamScore)

Next

End Sub

'------------------------------------------------------------

'- Subprogram Name: Main() -

'------------------------------------------------------------

'- Written By: Austin Rippee -

'- Written On: April 10, 2022 -

'------------------------------------------------------------

'- Subprogram Purpose: -

'- -

'- Main program that runs the application

'------------------------------------------------------------

'- Parameter Dictionary (in parameter order): -

'- (None)

'------------------------------------------------------------

'- Local Variable Dictionary (alphabetically): -

'- anExcelDoc - Instance of an excel application

'- dblFinalGrade - calculates the final grade

'- CheckExcel - checks if excel is already running

'- intCol - Keeps track of the current column

'- intRow - Keeps track of the current row

'- myStudents - a list which is an instance of clsStudent -

'------------------------------------------------------------

Sub Main()

'Sets the console title

Console.Title = "Students Overall Grades"

Console.Clear()

'Create new instance of clsStudent

Dim myStudents As New List(Of clsStudent)

'Adds students to the new list of class student

myStudents.Add(New clsStudent("V.A.", "Borstellis", {25, 25, 25, 25}, 100.0))

myStudents.Add(New clsStudent("A.S.", "Reid", {20, 21, 20, 18}, 75.0))

myStudents.Add(New clsStudent("C.U.", "Tyler", {19, 20, 21, 24}, 75.5))

myStudents.Add(New clsStudent("H.A.", "Renee", {20, 23, 23, 25}, 80.5))

myStudents.Add(New clsStudent("I.A.", "Douglas", {24, 23, 25, 25}, 95.0))

myStudents.Add(New clsStudent("M.A.", "Elenaips", {23, 24, 23, 21}, 94.5))

myStudents.Add(New clsStudent("A.L.", "Emmet", {21, 19, 18, 15}, 73.0))

myStudents.Add(New clsStudent("S.U.", "James", {21, 24, 23, 22}, 87.5))

myStudents.Add(New clsStudent("S.H.", "Issacs", {23, 24, 21, 21}, 93.0))

myStudents.Add(New clsStudent("B.I.", "Opus", {23, 24, 25, 23}, 97.5))

myStudents.Add(New clsStudent("T.R.", "Alski", {24, 25, 25, 23}, 95.5))

myStudents.Add(New clsStudent("H.E.", "Zeus", {23, 24, 23, 23}, 77.0))

myStudents.Add(New clsStudent("S.C.", "Ustaf", {24, 23, 24, 25}, 91.0))

myStudents.Add(New clsStudent("K.I.", "Chrint", {23, 23, 24, 21}, 89.0))

myStudents.Add(New clsStudent("J.E.", "Yaz", {25, 24, 23, 24}, 92.5))

myStudents.Add(New clsStudent("F.R.", "Franks", {23, 19, 18, 23}, 88.5))

myStudents.Add(New clsStudent("W.I.", "Walton", {24, 23, 23, 19}, 90.0))

myStudents.Add(New clsStudent("K.A.", "Gilch", {24, 23, 25, 24}, 92.0))

myStudents.Add(New clsStudent("R.O.", "Little", {23, 24, 23, 24}, 94.0))

myStudents.Add(New clsStudent("S.A.", "Xerxes", {24, 23, 25, 23}, 94.0))

myStudents.Add(New clsStudent("W.I.", "Harris", {23, 24, 25, 23}, 92.0))

myStudents.Add(New clsStudent("T.I.", "Vargo", {24, 23, 25, 25}, 99.0))

myStudents.Add(New clsStudent("I.E.", "Interas", {24, 23, 25, 25}, 97.5))

myStudents.Add(New clsStudent("T.O.", "Kiliens", {23, 19, 18, 18}, 73.0))

myStudents.Add(New clsStudent("E.R.", "Manrose", {23, 24, 25, 23}, 84.0))

myStudents.Add(New clsStudent("W.A.", "Nelson", {23, 24, 25, 23}, 87.0))

myStudents.Add(New clsStudent("K.U.", "Quaras", {23, 24, 25, 23}, 96.5))

Console.WriteLine("Displaying Students...")

Console.WriteLine("")

DisplayStudents(myStudents)

Console.WriteLine()

Console.WriteLine()

'Sets the default row and column to 1

Dim intRow As Integer = 1

Dim intCol As Integer = 1

'Initializes the excel application

Dim CheckExcel As Object = Nothing

Dim anExcelDoc As Excel.Application

'Check to see if Excel is already loaded in memory

Try

CheckExcel = GetObject(, "Excel.Application")

Catch Ex As Exception

'Excel was not running, so we got an error

End Try

If CheckExcel Is Nothing Then

'Create a new instance of Excel

anExcelDoc = New Excel.Application()

anExcelDoc.Visible = True

Else

anExcelDoc = CheckExcel

anExcelDoc.Visible = True

End If

Console.WriteLine("Opening Excel...")

Console.WriteLine("")

'Add a new workbook and a new sheet

anExcelDoc.Workbooks.Add()

'Sets the headers in specific cells

anExcelDoc.Cells(1, 1) = "Initials"

anExcelDoc.Cells(1, 2) = "Name"

anExcelDoc.Cells(1, 3) = "Grade 1"

anExcelDoc.Cells(1, 4) = "Grade 2"

anExcelDoc.Cells(1, 5) = "Grade 3"

anExcelDoc.Cells(1, 6) = "Grade 4"

anExcelDoc.Cells(1, 7) = "Grade Total"

anExcelDoc.Cells(1, 8) = "Exam"

anExcelDoc.Cells(1, 9) = "Final Grade"

Console.WriteLine("Column Titles Added.")

Console.WriteLine("")

'Sets the row to row 2 to get the program to know it wants to start on the row below the headers

intRow = 2

For Each student In myStudents

'Populates the cells with each of the students data

anExcelDoc.Cells(intRow, intCol) = student.getInitials

anExcelDoc.Cells(intRow, intCol + 1) = student.getLastName

anExcelDoc.Cells(intRow, intCol + 2) = student.getScores(0)

anExcelDoc.Cells(intRow, intCol + 3) = student.getScores(1)

anExcelDoc.Cells(intRow, intCol + 4) = student.getScores(2)

anExcelDoc.Cells(intRow, intCol + 5) = student.getScores(3)

anExcelDoc.Cells(intRow, intCol + 6) = "=SUM(" & getColumnLetter(intCol + 2) & intRow & ":" & getColumnLetter(intCol + 5) & intRow

anExcelDoc.Cells(intRow, intCol + 7) = student.getExamScore

'Gets the final grade value

Dim dblFinalGrade As Double = ((student.getScores(0) + student.getScores(1) + student.getScores(2) + student.getScores(3)) \* 0.4) + (student.getExamScore \* 0.6)

'Displays the final grade value

anExcelDoc.Cells(intRow, intCol + 8) = "=" & dblFinalGrade

'Increases the row number

intRow += 1

Next

'Creates headers for the average, stddev, min, and max

anExcelDoc.Cells(intRow + 1, intCol + 1) = "Aver:"

anExcelDoc.Cells(intRow + 2, intCol + 1) = "St Dev:"

anExcelDoc.Cells(intRow + 3, intCol + 1) = "Min:"

anExcelDoc.Cells(intRow + 4, intCol + 1) = "Max:"

'Creates column for every average

anExcelDoc.Cells(intRow + 1, intCol + 2) = "=AVERAGE(" & getColumnLetter(intCol + 2) & (intRow - intRow + 2) & ":" & getColumnLetter(intCol + 2) & (intRow - 1) & ")"

anExcelDoc.Cells(intRow + 1, intCol + 3) = "=AVERAGE(" & getColumnLetter(intCol + 3) & (intRow - intRow + 2) & ":" & getColumnLetter(intCol + 3) & (intRow - 1) & ")"

anExcelDoc.Cells(intRow + 1, intCol + 4) = "=AVERAGE(" & getColumnLetter(intCol + 4) & (intRow - intRow + 2) & ":" & getColumnLetter(intCol + 4) & (intRow - 1) & ")"

anExcelDoc.Cells(intRow + 1, intCol + 5) = "=AVERAGE(" & getColumnLetter(intCol + 5) & (intRow - intRow + 2) & ":" & getColumnLetter(intCol + 5) & (intRow - 1) & ")"

anExcelDoc.Cells(intRow + 1, intCol + 6) = "=AVERAGE(" & getColumnLetter(intCol + 6) & (intRow - intRow + 2) & ":" & getColumnLetter(intCol + 6) & (intRow - 1) & ")"

anExcelDoc.Cells(intRow + 1, intCol + 7) = "=AVERAGE(" & getColumnLetter(intCol + 7) & (intRow - intRow + 2) & ":" & getColumnLetter(intCol + 7) & (intRow - 1) & ")"

anExcelDoc.Cells(intRow + 1, intCol + 8) = "=AVERAGE(" & getColumnLetter(intCol + 8) & (intRow - intRow + 2) & ":" & getColumnLetter(intCol + 8) & (intRow - 1) & ")"

'Creates column for every standard deviation

anExcelDoc.Cells(intRow + 2, intCol + 2) = "=STDEVA(" & getColumnLetter(intCol + 2) & (intRow - intRow + 2) & ":" & getColumnLetter(intCol + 2) & (intRow - 1) & ")"

anExcelDoc.Cells(intRow + 2, intCol + 3) = "=STDEVA(" & getColumnLetter(intCol + 3) & (intRow - intRow + 2) & ":" & getColumnLetter(intCol + 3) & (intRow - 1) & ")"

anExcelDoc.Cells(intRow + 2, intCol + 4) = "=STDEVA(" & getColumnLetter(intCol + 4) & (intRow - intRow + 2) & ":" & getColumnLetter(intCol + 4) & (intRow - 1) & ")"

anExcelDoc.Cells(intRow + 2, intCol + 5) = "=STDEVA(" & getColumnLetter(intCol + 5) & (intRow - intRow + 2) & ":" & getColumnLetter(intCol + 5) & (intRow - 1) & ")"

anExcelDoc.Cells(intRow + 2, intCol + 6) = "=STDEVA(" & getColumnLetter(intCol + 6) & (intRow - intRow + 2) & ":" & getColumnLetter(intCol + 6) & (intRow - 1) & ")"

anExcelDoc.Cells(intRow + 2, intCol + 7) = "=STDEVA(" & getColumnLetter(intCol + 7) & (intRow - intRow + 2) & ":" & getColumnLetter(intCol + 7) & (intRow - 1) & ")"

anExcelDoc.Cells(intRow + 2, intCol + 8) = "=STDEVA(" & getColumnLetter(intCol + 8) & (intRow - intRow + 2) & ":" & getColumnLetter(intCol + 8) & (intRow - 1) & ")"

'Creates column for every min

anExcelDoc.Cells(intRow + 3, intCol + 2) = "=MIN(" & getColumnLetter(intCol + 2) & (intRow - intRow + 2) & ":" & getColumnLetter(intCol + 2) & (intRow - 1) & ")"

anExcelDoc.Cells(intRow + 3, intCol + 3) = "=MIN(" & getColumnLetter(intCol + 3) & (intRow - intRow + 2) & ":" & getColumnLetter(intCol + 3) & (intRow - 1) & ")"

anExcelDoc.Cells(intRow + 3, intCol + 4) = "=MIN(" & getColumnLetter(intCol + 4) & (intRow - intRow + 2) & ":" & getColumnLetter(intCol + 4) & (intRow - 1) & ")"

anExcelDoc.Cells(intRow + 3, intCol + 5) = "=MIN(" & getColumnLetter(intCol + 5) & (intRow - intRow + 2) & ":" & getColumnLetter(intCol + 5) & (intRow - 1) & ")"

anExcelDoc.Cells(intRow + 3, intCol + 6) = "=MIN(" & getColumnLetter(intCol + 6) & (intRow - intRow + 2) & ":" & getColumnLetter(intCol + 6) & (intRow - 1) & ")"

anExcelDoc.Cells(intRow + 3, intCol + 7) = "=MIN(" & getColumnLetter(intCol + 7) & (intRow - intRow + 2) & ":" & getColumnLetter(intCol + 7) & (intRow - 1) & ")"

anExcelDoc.Cells(intRow + 3, intCol + 8) = "=MIN(" & getColumnLetter(intCol + 8) & (intRow - intRow + 2) & ":" & getColumnLetter(intCol + 8) & (intRow - 1) & ")"

'Creates column for every max

anExcelDoc.Cells(intRow + 4, intCol + 2) = "=MAX(" & getColumnLetter(intCol + 2) & (intRow - intRow + 2) & ":" & getColumnLetter(intCol + 2) & (intRow - 1) & ")"

anExcelDoc.Cells(intRow + 4, intCol + 3) = "=MAX(" & getColumnLetter(intCol + 3) & (intRow - intRow + 2) & ":" & getColumnLetter(intCol + 3) & (intRow - 1) & ")"

anExcelDoc.Cells(intRow + 4, intCol + 4) = "=MAX(" & getColumnLetter(intCol + 4) & (intRow - intRow + 2) & ":" & getColumnLetter(intCol + 4) & (intRow - 1) & ")"

anExcelDoc.Cells(intRow + 4, intCol + 5) = "=MAX(" & getColumnLetter(intCol + 5) & (intRow - intRow + 2) & ":" & getColumnLetter(intCol + 5) & (intRow - 1) & ")"

anExcelDoc.Cells(intRow + 4, intCol + 6) = "=MAX(" & getColumnLetter(intCol + 6) & (intRow - intRow + 2) & ":" & getColumnLetter(intCol + 6) & (intRow - 1) & ")"

anExcelDoc.Cells(intRow + 4, intCol + 7) = "=MAX(" & getColumnLetter(intCol + 7) & (intRow - intRow + 2) & ":" & getColumnLetter(intCol + 7) & (intRow - 1) & ")"

anExcelDoc.Cells(intRow + 4, intCol + 8) = "=MAX(" & getColumnLetter(intCol + 8) & (intRow - intRow + 2) & ":" & getColumnLetter(intCol + 8) & (intRow - 1) & ")"

Console.WriteLine("Information displayed")

'Sizes each row to fit to data

anExcelDoc.Range("A:I").EntireColumn.AutoFit()

'Cleans things up

anExcelDoc.Quit()

anExcelDoc = Nothing

Console.ReadLine()

End Sub

'------------------------------------------------------------

'- Function Name: getColumnLetter -

'------------------------------------------------------------

'- Written By: Austin Rippee

'- Written On: April 10, 2022 -

'------------------------------------------------------------

'- Subprogram Purpose: -

'- -

'- Gtes the string letter value of what number it correlates to

'------------------------------------------------------------

'- Parameter Dictionary (in parameter order): -

'- colNumber - Number attempting to convert

'------------------------------------------------------------

'- Local Variable Dictionary (alphabetically): -

'- (None) -

'------------------------------------------------------------

'- Returns: -

'- String – letter representation -

'------------------------------------------------------------

Public Function getColumnLetter(ByVal colNumber As Integer) As String

If colNumber = 1 Then

Return "A"

ElseIf colNumber = 2 Then

Return "B"

ElseIf colNumber = 3 Then

Return "C"

ElseIf colNumber = 4 Then

Return "D"

ElseIf colNumber = 5 Then

Return "E"

ElseIf colNumber = 6 Then

Return "F"

ElseIf colNumber = 7 Then

Return "G"

ElseIf colNumber = 8 Then

Return "H"

ElseIf colNumber = 9 Then

Return "I"

ElseIf colNumber = 10 Then

Return "J"

ElseIf colNumber = 11 Then

Return "K"

ElseIf colNumber = 12 Then

Return "L"

ElseIf colNumber = 13 Then

Return "M"

ElseIf colNumber = 14 Then

Return "N"

ElseIf colNumber = 15 Then

Return "O"

ElseIf colNumber = 16 Then

Return "P"

ElseIf colNumber = 17 Then

Return "Q"

ElseIf colNumber = 18 Then

Return "R"

ElseIf colNumber = 19 Then

Return "S"

ElseIf colNumber = 20 Then

Return "T"

ElseIf colNumber = 21 Then

Return "U"

ElseIf colNumber = 22 Then

Return "V"

ElseIf colNumber = 23 Then

Return "W"

ElseIf colNumber = 24 Then

Return "X"

ElseIf colNumber = 25 Then

Return "Y"

ElseIf colNumber = 26 Then

Return "Z"

Else

Return " "

End If

End Function

End Module

Table

Description automatically generated