```
Public Class wordGraph
    Private word As String ' word that we seach about
    Private GraphTree As Term ' the tree of word and related words
    Private visits As New Dictionary(Of String, Boolean)
    Public Sub New(ByVal word As String, ByVal engFlag As Boolean)
        Me.word = word
        Me.GraphTree = New Term(word, engFlag)
    End Sub

    ' this function used to generate word graph
    Public Sub buildTree(ByVal maxDepth As Integer)
        ' call buildTree with the intial word
        buildTree(Me.GraphTree, maxDepth, "")
    End Sub

    Public Function FindCircleGraph() As List(Of String)
        Dim stack As New Stack
        Dim ResStack As New List(Of String)
        Me.FindCircleGraph(Me.GraphTree, Me.GraphTree.VALUE, stack, ResStack, "")
        Return ResStack
    End Function


    Public Sub FindCircleGraph(ByRef node As Term, ByVal intialWord As String, ByRef stack As Stack,↙
     ByRef ResStack As List(Of String), ByVal prev As String)
        If node.getNumberOfLinks >= 2 Then
            If stack.Contains(node.VALUE) And node.VALUE = intialWord Then
                For Each item As String In stack
                    If Not ResStack.Contains(item) Then
                        ResStack.Add(item)
                    End If
                Next
            End If

            If Not stack.Contains(node.VALUE) Then
                stack.Push(node.VALUE)
                For Each subNode As Term In node.LINKS
                    If (Not subNode.VALUE.ToLower = prev.ToLower) Then
                        FindCircleGraph(subNode, intialWord, stack, ResStack, node.VALUE)
                    End If
                Next
                stack.Pop()
            End If

        End If
    End Sub


    Private Sub printlist(ByRef s As Stack)
        Dim r As String = ""
        For Each i As String In s
            r = r + i + ", " + Environment.NewLine
        Next
        MsgBox(r)
    End Sub

    Private Function getV(ByVal x As String) As Boolean
        If Me.visits.ContainsKey(x) Then
            Return Me.visits(x)
        Else
            Return 333
        End If

    End Function

    ' implementation of build graph function
    Public Sub buildTree(ByRef node As Term, ByVal maxDepth As Integer, ByVal prev As String)
        Dim value As String = ""
        If maxDepth >= 0 Then ' loop until reach depth zero

            Dim adp As New thesaurusDataSetTableAdapters.testTableAdapter
            Dim tbl As DataTable
```

```vb
            If node.ENGLISH_FLAG Then
                ' if the term is english, get all arabic words related to this term
                tbl = adp.GetDataByEnglish(node.VALUE)
            Else
                ' if the term is arabic, get all english words related to this term
                tbl = adp.GetDataByArabic(node.VALUE)
            End If


            For Each row As DataRow In tbl.Rows
                ' build new term until term is already exist
                value = row.ItemArray(If(node.ENGLISH_FLAG, 2, 1)).ToString
                If Not value.ToLower = prev.ToLower Then
                    Dim subTerm As New Term(value, Not node.ENGLISH_FLAG)
                    node.addAdjacent(subTerm)
                    subTerm.addAdjacent(node)
                    Me.buildTree(subTerm, maxDepth - 1, node.VALUE)
                End If

            Next

        End If
        ' code for building tree should be here
        ' when the node is completely built we remove it from the stack

    End Sub

    Public ReadOnly Property TREE() As Term
        Get
            Return Me.GraphTree
        End Get
    End Property

End Class

```

```vb
Public Class Term
    Private engFlag As Boolean
    Private _value As String ' here we save the term
    Public _links As List(Of Term) ' we store term adj
    Private _visited As Boolean = False
    Public Sub New(ByVal value As String, ByVal engFlag As Boolean)
        Me.engFlag = engFlag
        Me._links = New List(Of Term)
        Me._value = value
    End Sub

    Public Sub addAdjacent(ByRef term As Term)
        Me._links.Add(term)
    End Sub

    Public Sub removeLink(ByRef key As String)
        For Each link As Term In Me._links
            If key = link.VALUE Then
                link = Nothing
                Exit For
            End If

        Next
    End Sub
    Public Function checkAjacentNodeIfExist(ByRef term As Term) As Boolean
        For Each t As Term In Me.LINKS
            If term Is t Then
                Return True
            End If
        Next
        Return False
    End Function
    Public ReadOnly Property ENGLISH_FLAG() As Boolean
        Get
            Return Me.engFlag
        End Get
    End Property

    Public Property VALUE() As String
        Get
            Return Me._value
        End Get
        Set(ByVal value As String)
            Me._value = value
        End Set
    End Property

    Public Property VISITED() As Boolean
        Get
            Return Me._visited
        End Get
        Set(ByVal value As Boolean)
            Me._visited = value
        End Set
    End Property

    Public ReadOnly Property LINKS() As List(Of Term)
        Get
            Return Me._links
        End Get
    End Property

    Public Function getNumberOfLinks() As Integer
        Try
            Return Me._links.Count
        Catch ex As Exception
            Return 0
        End Try
    End Function
End Class
```