### Deeply Recursive Ackermann Function with Memoization

Wilhelm Ackermann explored recursive functions, such as Fibonacci, but looked further at much more complicated, potentially doubly recursive functions. He invented the Ackermann Function, which grows memory and stack use (much!) faster than exponential. More recently, people have sped up the evaluation of the Ackermann function by memoization. Recognizing that such a deeply recursive algorithm often covers the same evaluations many, many times, memoization stores evaluated results, and allows a non-naive algorithm to determine whether further recursion is needed. This mmBasic implementation uses memoization to evaluate the Ackermann function. Because of the design of the function, evaluation grows much faster “vertically” than “horizontally”. The Ackermann memoization array is defined five times deeper than it is wide. mmBasic's own implementation limits recursive calls to ~100 deep. Once it gets that deep, the program will crash.

```
'
************************************************************************\
\
'   Deeply Recursive Ackermann Sequence using Memoization in MMBasic
'   See https://en.wikipedia.org/wiki/Ackermann_function for more
information
'   Steven F. Johnson                                          September
2020
'
'   A(m, n):  m = 0        A(0, n) = n+1                (No recursion    )
'             m > 0, n = 0  A(m, 0) = A(m-1, 1)           (Single recursion)
'             Otherwise     A(m, n) = A(m-1, A(m, n-1))  (Double recursion)
'
************************************************************************

' **********  Initialization Section
************************************
Option BASE 0 : OPTION EXPLICIT
Dim INTEGER i, j, MemAck(20,1000)' Allocate space for Results

For i=0 to 19
  For j=0 to 999
    MemAck(i,j)=-99  ' indicate that this is not yet evaluated
  Next j
Next i

' **********   Function Definitions
************************************
Function Ack(m, n) As INTEGER ' Implements Ackermann Function

  If (m=0) Then                        ' Simplest case - no recursion
    MemAck(m,n) = n+1                  ' Memoize it!
  ElseIf ((m>0) And (n=0)) Then        ' Medium case - single recursion
    If MemAck(m-1,1) < 0 Then          ' Check to see if value already there
      MemAck(m,n) = Ack(m-1,1)         ' Calculate, then Memoize it
    Else
      MemAck(m,n) = MemAck(m-1,1)      ' Memoize the existing value
```

```
    EndIf

  Else                                  ' Most complicated - potential double
recursion
    If MemAck(m, n-1) < 0 Then MemAck(m, n-1) = Ack(m, n-1)
' See if Right Hand already evaluated
    If MemAck(m-1, MemAck(m, n-1)) < 0 Then MemAck(m-1, MemAck(m, n-1)) =
Ack(m-1, MemAck(m, n-1)) ' Check for Left Hand value
    MemAck(m,n) = MemAck(m-1,MemAck(m, n-1))
' Memoize it!

  Endif

  Ack = MemAck(m,n)                     ' Set return value for function to
memoized value

End Function

' **********   Main Body of Program
**************************************

CLS
Print "Press Ctrl-C to interrupt execution"

For i = 0 to 9
  For j = 0 to 9
    Print "   Ack("; i; ", "; j; " ): "; Ack(i,j)
  Next j
Next i

End
```