Pseudocode for a recursive solution to the Knight's Tour problem.

The players:

- o Let **N** be the size of the board (**N** × **N**).
- o Let **M** = $\mathbf{N}^2$ – 1. This is the number of valid moves the knight must make to complete a tour.
- o Let **visited** be a two dimensional, **N** × **N**, array of Boolean values. This array is initialized to false, and a location, (x,y), in the array is set to true when the knight visits that spot.
- o Let **m** be an integer between 0 and **M**. It represents a move number. The initial location of the knight is **m** = 0 and, when the knight has completed a tour, **m** = **M**.

```
Boolean Algorithm Move( x, y, m )
// (x, y) is a location of the board and m is a move number

   if( (x < 0) OR (x ≥ N) OR (y < 0) OR (y ≥ N) )
      return false //A coordinate is off the board

   if( visited[x, y] = true )
      return false //Can't move here; it has already been visited

   if( m = M )
      //This is a valid move and the knight has now made M moves; so,
      //we have a solution!!!
      print "A solution has been found"
      print " x, y " //This starts printing the solution
      set visited[x, y] = true
      return true
   else
      //This is a valid move, but a tour has not been completed.
      //So, try all the moves that can be made from this location
      //recursively.

      let result be a Boolean variable  //MUST be local
      set result = false

      set result = result OR Move( x+2, y+1, m+1)
      set result = result OR Move( x+2, y-1, m+1)
      set result = result OR Move( x-2, y+1, m+1)
      set result = result OR Move( x-2, y-1, m+1)
      set result = result OR Move( x+1, y+2, m+1)
      set result = result OR Move( x+1, y-2, m+1)
      set result = result OR Move( x-1, y+2, m+1)
      set result = result OR Move( x-1, y-2, m+1)
```

```
    if( result = true )
       //One of the 8 moves above led to a completed tour.  So, this
       //position is part of a successful tour.
       print " x, y "
       return true
    else
       //None of the moves from this position led to a successful
       //tour.  Now we must backtrack and try a different path
       set visited[x, y] = false //Unvisit this location
       return false
```

To find a tour, for example, starting from (3, 3) on a 5 × 5 board, the initial call would be

```
    Move( 3, 3, 0 )
```

If this call returns `true`, a tour is found and the solution is printed in reverse order. If the call returns `false`, no solution was found.