Use the iteration method to solve the following recurrence relations:

1. \( T(n) = 3T(n/4) + n \)
2. \( T(n) = T(n-2) + 1 \)
3. \( T(n) = 2T(n/2) + n \)
4. \( T(n) = 3T(n-1) + n^4 \)
5. \( T(n) = 16T(n/4) + n^2 \)
6. \( T(n) = 7T(n/2) + n^2 \)

7. procedure recursive(n: integer);
   begin
     if \( n \leq 1 \) then
       return(1);
     else
       return(recursive(n-1) + recursive(n-1));
   end;

8. Algorithm recursiveMax(A, n):
   Input: An array \( A \) storing \( n \geq 1 \) integers.
   Output: The maximum element in \( A \).
   if \( n = 1 \) then
     return \( A[0] \)
   return \( \max(\text{recursiveMax}(A, n-1), A[n-1]) \)
*(note: assume that \( \max(x, y) \) is a function that takes 2 integers and returns the largest in constant time.)*

If you have time, do an analysis on these:

procedure lessTricky(n)
  \( i \leftarrow 1 \)
  while \( i < n \) do
    \( n = n/2 \)
  return \( n \)

procedure loops(n)
  \( count = 0 \)
  for \( i = 1 \) to \( n \) do
    for \( j = 1 \) to \( i \) do
      \( count = count + 3 \)
  return \( count \)