Example 8

The following code:

```
(define (T3 N)
  (cond [(not (integer? N)) 'expect-an-integer]
        [(< N 1)            'expect-1-or-higher]
        [(= N 1)   1]
        [else (+(fac N)
                 (T3 (- N 1)))]))
```

Has this recurrence relationship:

```
T(1) = 1
T(N) = 1 + N + T(N-1) for all N >1
```

Cheat sheet (rewritten versions of the recurrence relationship):

```
T(N-1) = N + T(N-2) #equation 2
T(N-2) = N-1 + T(N-3) #equation 3
T(N-3) = N-2 + T(N-4) #equation 4
```

Recurrence Relationship Unrolling

```
T(N) = 1 + N + T(N-1)

Plugging in equation 2:
T(N) = 2 + N + N-1 + T(N-2)

Plugging in equation 3:
T(N) = 3 + N + N-1 + N-2 + T(N-3)

Plugging in equations 4:
T(N) = 4 + N + N-1 + N-2 + N-3 + T(N-4)

We see a pattern here! We figure out the sum by plugging in a few small
     values for N to see what sum we should get. From that we can
     generalize to the following sum:

This will result in the following sum:
T(N) = (N – 1) + N + N-1 + N-2 + ... + 4 + 3 + 2 + 1


The sum of that series is N(N+1)/2 + (N-1)
Therefore the Big-O runtime is:
O(N^2)
```