Assignment 2 part 3
Recurrence Relation

Step through first

```
function mergeSort(x)
{let tmp = [ ] → 1
  for( i=0  i<x.len  ++i) → N
  {
      Tmp.unshift([i, i]) → 1
  }

  while(tmp.length > 2) → n
{
  [alo, aHi], [bLo, BHi] = [ tmp.pop, tmp.pop] → 1
  if alo === blo  && aHi == BHi → 1
      unshift[alo, aHi] → 1
      continue
    lo = min(alo, blo) → 1
  mid = aLo < bLo ? aHi : bHi → 1
  hi = max(aHi, bHi) → 1
    if (lo > Hi) → 1
     continue

merge inplace(x, lo, mid, Hi) → n
    unshift(lo, hi] → 1

  }
```

So we have

$T(n) = 1 + N + 1 + N + 1 + 1 + 1 + 1 + 1 + 1 + 1 + N + 1$
$= 3n + 10$

$T(n) = \Theta(n)$

for the tight bounds on my merge sort function we would have $\Theta(n)$. Everything is only done as long as n is. Where the function is not recursive, you don't have a difficult recurrence relation.