```
#The pseudocode for the algorithm of long division square root calculation
#given function takes all integer arguments

function findNextNum (divisor10, remainder, current)
{
      if (current == 10) then return 9;
      value ← (divisor10*current + current*current);
      if(value <= remainder)
            return findNextNum (divisor10, remainder, current+1);
      else
            return (current-1);
       #Returns the largest value that satisfies the inequality.

}

function SQRT(quotient, divisor, x, remainder)
{
      #x can be divided into groups of 2, and if it had odd number of
      # digits, assume it has a leading 0

      if (number == 0) then
             return (quotient, remainder) as a pair of integers;
             #base case of recursion
      else
            ff ← x[1,2];
            #the first two digits of x given as one decimal number

            x ← x[2..size(x)]; #remove the first two digits from x
            remainder ← remainder*100 + ff;
            #the new remainder after bringing the next group to the right

            nextDigit ← findNextNum(divisor*10, remainder,0)
            #returns largest digit(0-9) c such that (divisor*10*c + c^2 <= remainder);

            remainder ← (remainder - (divisor*10 + nextDigit)*10);
            #remainder after subtraction

            quotient ← 10*quotient + nextDigit; #added digit to end of quotient
            divisor ← divisor*10  + 2*nextDigit;

        return SQRT(quotient, divisor, x, remainder) #tail recursive step;

}

Function isqrtld(x)
{
      Return SQRT(0,0,x,0);
}
```