NumToText

# Description of the problem:

Create code to take a number input and return the equivalent in British English words e.g.

1 = one

21 = twenty one

105 = one hundred and five

56,945,781 = fifty six million, nine hundred and forty five thousand, seven hundred and eighty one

etc.. up to 999,999,999

# Approach:

* First identify the repetition of the numbers in words
  + From 1 to 9 in words
  + From ten, eleven, thirteen, fourteen, fifteen,sixteen,seventeen,eigthteen,nineteen
  + From 20 to 90
  + 100 – hundred
  + 1000 – thousand
  + 1000000 – million
* Then get the length of the number
* Split the number into digits and put them in a list
* Map string values to int values
  + 1 -> one digit -> number
  + 42 -> two digits -> (number + number) or (number)
  + 245 -> three digits -> number + hundred + ‘and’ + either one number word or two
  + 4,564 -> four digits -> number + thousand + ‘and’ + number + hundred + ‘and’ + two digits
  + 53,444 -> five digits -> either one number word or two + thousand + ‘and’ + number + hundred + ‘and’ + either one number word or two
  + 456,456 -> six digits -> number + hundred + ‘and’ + either one number word or two + thousand +’and’ + number + hundred + either one number word or two
  + 7,894,561 -> seven digits -> number + million + ‘,’ + number + hundred + ‘and’ + either one number word or two + thousand + ‘,’ + number + hundred + ‘and’ + either one number word or two
  + 78,945,612 -> eight digits -> either one number word or two + million +’,’ +number + hundred + ‘and’ + either one number word or two + thousand + ‘,’ + number + hundred + ‘and’ + either one number word or two
  + 999,999,999 -> nine digits -> number + hundred + ‘and’ + either one number word or two+ million + ‘,’ + number + hundred + either one number word or two + thousand + ‘,’ + number + hundred + ‘and’ + either one number word or two

Print numbers :

1 parseDecimalOneDigitNumberToText(*List*(number.head)) + ConjunctionAndOthers.*space* +Hundred.*hundred*

2 parseDecimalOneDigitNumberToText(*List*(number.head)) + ConjunctionAndOthers.*space* +Hundred.*hundred*

ConjunctionAndOthers.*space* + ConjunctionAndOthers.*and* + ConjunctionAndOthers.*space* +  
 parseDecimalTwoDigitNumberToText(restOfNumberAsInt)

3 parseDecimalOneDigitNumberToText(*List*(number.head)) + ConjunctionAndOthers.*space* +  
 Thousand.*thousand*

4 parseDecimalOneDigitNumberToText(*List*(number.head)) + ConjunctionAndOthers.*space* +  
 Thousand.*thousand* + ConjunctionAndOthers.*space* + ConjunctionAndOthers.*and +*

ConjunctionAndOthers.*space* + parseHundredNumberToText(number.tail)

5 parseDecimalTwoDigitNumberToText(splittedList.\_1) + ConjunctionAndOthers.*space* +  
 Thousand.*thousand*

6 parseDecimalTwoDigitNumberToText(splittedList.\_1) + ConjunctionAndOthers.*space* +  
 Thousand.*thousand* + ConjunctionAndOthers.*space* + ConjunctionAndOthers.*and* + ConjunctionAndOthers.*space +* parseHundredNumberToText(splittedList.\_2)

7 parseDecimalTwoDigitNumberToText(splittedList.\_1) + ConjunctionAndOthers.*space* +   
 Thousand.*thousand* + ConjunctionAndOthers.*comma* + ConjunctionAndOthers.*space +* parseHundredNumberToText(splittedList.\_2)

8 parseHundredNumberToText(splittedNumber.\_1) + ConjunctionAndOthers.*space* + Thousand.*thousand*

9 parseHundredNumberToText(splittedNumber.\_1) + ConjunctionAndOthers.*space* + Thousand.*thousand* +  
 ConjunctionAndOthers.*space* + ConjunctionAndOthers.*and* + ConjunctionAndOthers.*space +*

parseHundredNumberToText(splittedNumber.\_2)

10 parseHundredNumberToText(splittedNumber.\_1) + ConjunctionAndOthers.*space* + Thousand.*thousand* + ConjunctionAndOthers.*comma +* ConjunctionAndOthers.*space* +parseHundredNumberToText(splittedNumber.\_2)