```python
from __future__ import print_function

american_number_system = {
    'zero': 0,
    'one': 1,
    'two': 2,
    'three': 3,
    'four': 4,
    'five': 5,
    'six': 6,
    'seven': 7,
    'eight': 8,
    'nine': 9,
    'ten': 10,
    'eleven': 11,
    'twelve': 12,
    'thirteen': 13,
    'fourteen': 14,
    'fifteen': 15,
    'sixteen': 16,
    'seventeen': 17,
    'eighteen': 18,
    'nineteen': 19,
    'twenty': 20,
    'thirty': 30,
    'forty': 40,
    'fifty': 50,
    'sixty': 60,
    'seventy': 70,
    'eighty': 80,
    'ninety': 90,
    'hundred': 100,
    'thousand': 1000,
    'million': 1000000,
    'billion': 1000000000,
    'point': '.'
}

decimal_words = ['zero', 'one', 'two', 'three', 'four', 'five', 'six', 'seven',
'eight', 'nine']

"""
#TODO
indian_number_system = {
    'zero': 0,
    'one': 1,
    'two': 2,
    'three': 3,
    'four': 4,
    'five': 5,
    'six': 6,
    'seven': 7,
    'eight': 8,
    'nine': 9,
    'ten': 10,
    'eleven': 11,
    'twelve': 12,
    'thirteen': 13,
    'fourteen': 14,
    'fifteen': 15,
    'sixteen': 16,
    'seventeen': 17,
    'eighteen': 18,
    'nineteen': 19,
    'twenty': 20,
    'thirty': 30,
    'forty': 40,
    'fifty': 50,
    'sixty': 60,
    'seventy': 70,
    'eighty': 80,
    'ninety': 90,
```

```
    'hundred': 100,
    'thousand': 1000,
    'lac': 100000,
    'lakh': 100000,
    'crore': 10000000
}
"""


"""
function to form numeric multipliers for million, billion, thousand etc.
input: list of strings
return value: integer
"""


def number_formation(number_words):
    numbers = []
    for number_word in number_words:
        numbers.append(american_number_system[number_word])
    if len(numbers) == 4:
        return (numbers[0] * numbers[1]) + numbers[2] + numbers[3]
    elif len(numbers) == 3:
        return numbers[0] * numbers[1] + numbers[2]
    elif len(numbers) == 2:
        if 100 in numbers:
            return numbers[0] * numbers[1]
        else:
            return numbers[0] + numbers[1]
    else:
        return numbers[0]


"""
function to convert post decimal digit words to numerial digits
input: list of strings
output: double
"""


def get_decimal_sum(decimal_digit_words):
    decimal_number_str = []
    for dec_word in decimal_digit_words:
        if(dec_word not in decimal_words):
            return 0
        else:
            decimal_number_str.append(american_number_system[dec_word])
    final_decimal_string = '0.' + ''.join(map(str,decimal_number_str))
    return float(final_decimal_string)


"""
function to return integer for an input `number_sentence` string
input: string
output: int or double or None
"""


def word_to_num(number_sentence):
    if type(number_sentence) is not str:
        raise ValueError("Type of input is not string! Please enter a valid number
        word (eg. \'two million twenty three thousand and forty nine\')")

    number_sentence = number_sentence.replace('-', ' ')
    number_sentence = number_sentence.lower()  # converting input to lowercase

    if(number_sentence.isdigit()):  # return the number if user enters a number string
        return int(number_sentence)

    split_words = number_sentence.strip().split()  # strip extra spaces and split
    sentence into words

```

```
142         clean_numbers = []
143         clean_decimal_numbers = []
144 
145         # removing and, & etc.
146         for word in split_words:
147             if word in american_number_system:
148                 clean_numbers.append(word)
149 
150         # Error message if the user enters invalid input!
151         if len(clean_numbers) == 0:
152             raise ValueError("No valid number words found! Please enter a valid number
                word (eg. two million twenty three thousand and forty nine)")
153 
154         # Error if user enters million,billion, thousand or decimal point twice
155         if clean_numbers.count('thousand') > 1 or clean_numbers.count('million') > 1 or
            clean_numbers.count('billion') > 1 or clean_numbers.count('point')> 1:
156             raise ValueError("Redundant number word! Please enter a valid number word
                (eg. two million twenty three thousand and forty nine)")
157 
158         # separate decimal part of number (if exists)
159         if clean_numbers.count('point') == 1:
160             clean_decimal_numbers = clean_numbers[clean_numbers.index('point')+1:]
161             clean_numbers = clean_numbers[:clean_numbers.index('point')]
162 
163         billion_index = clean_numbers.index('billion') if 'billion' in clean_numbers
            else -1
164         million_index = clean_numbers.index('million') if 'million' in clean_numbers
            else -1
165         thousand_index = clean_numbers.index('thousand') if 'thousand' in clean_numbers
            else -1
166 
167         if (thousand_index > -1 and (thousand_index < million_index or thousand_index <
            billion_index)) or (million_index>-1 and million_index < billion_index):
168             raise ValueError("Malformed number! Please enter a valid number word (eg.
                two million twenty three thousand and forty nine)")
169 
170         total_sum = 0  # storing the number to be returned
171 
172         if len(clean_numbers) > 0:
173             # hack for now, better way TODO
174             if len(clean_numbers) == 1:
175                     total_sum += american_number_system[clean_numbers[0]]
176 
177             else:
178                 if billion_index > -1:
179                     billion_multiplier = number_formation(clean_numbers[0:billion_index])
180                     total_sum += billion_multiplier * 1000000000
181 
182                 if million_index > -1:
183                     if billion_index > -1:
184                         million_multiplier =
                            number_formation(clean_numbers[billion_index+1:million_index])
185                     else:
186                         million_multiplier =
                            number_formation(clean_numbers[0:million_index])
187                     total_sum += million_multiplier * 1000000
188 
189                 if thousand_index > -1:
190                     if million_index > -1:
191                         thousand_multiplier =
                            number_formation(clean_numbers[million_index+1:thousand_index])
192                     elif billion_index > -1 and million_index == -1:
193                         thousand_multiplier =
                            number_formation(clean_numbers[billion_index+1:thousand_index])
194                     else:
195                         thousand_multiplier =
                            number_formation(clean_numbers[0:thousand_index])
196                     total_sum += thousand_multiplier * 1000
197 
198                 if thousand_index > -1 and thousand_index != len(clean_numbers)-1:
199                     hundreds = number_formation(clean_numbers[thousand_index+1:])
200                 elif million_index > -1 and million_index != len(clean_numbers)-1:
```

```
                hundreds = number_formation(clean_numbers[million_index+1:])
            elif billion_index > -1 and billion_index != len(clean_numbers)-1:
                hundreds = number_formation(clean_numbers[billion_index+1:])
            elif thousand_index == -1 and million_index == -1 and billion_index == -1:
                hundreds = number_formation(clean_numbers)
            else:
                hundreds = 0
            total_sum += hundreds

    # adding decimal part to total_sum (if exists)
    if len(clean_decimal_numbers) > 0:
        decimal_sum = get_decimal_sum(clean_decimal_numbers)
        total_sum += decimal_sum

    return total_sum
```