Write a function that combines two lists by alternatingly taking elements, e.g.

[a,b,c], [1,2,3] → [a,1,b,2,c,3].

[1,2,5,8,0], [9,4,8,7,6] → [1, 9, 2, 4, 5, 8, 8, 7, 0, 6].

Sol

[١,٢,٣] =m

[٤,٥,٦] =n

[٧,٨,٩] =o

*:Def. my Fun (*x)*

[] =s

:For i in x

s += i

Return s

Print my Fun (m,n,o) # [1, 2, 3, 4, 5, 6, 7, 8, 9] This is concatenating.

2.

n[[٩ ،٨ ،٧ ،٦ ،٥ ،٤] ،[٣ ،٢ ،١]] =

```
def flatten(*lsts):
    nlst[]=
    for i in lsts:
        for j in i:
            nlst+=j
    return nlst
```

Write a program that accepts an array of numbers and returns an array of numbers in written form e.g.

“] → [١,٤,٦]one”, “four”, “six[”

“] → [٠,٠,٦,٢,٧]zero”, “zero”, “six”,"two","seven["

[5,4,3,2,1,5,8] → ["five", "four", "three", "two", "one", "five","eight"]

print flatten(n)

sol

```
var IS_SOUTH_ASIAN = true;

function int_to_words(int){

  if (int === 0) return 'zero;'


  var ONES_WORD  =
['','one','two','three','four','five','six','seven','eight','nine','ten','eleven',
'twelve','thirteen','fourteen','fifteen','sixteen','seventeen','eighteen','nin
eteen'];

  var TENS_WORD  =
['','','twenty','thirty','fourty','fifty','sixty','seventy','eighty','ninety'];

  var SCALE_WORD_WESTERN =
['','thousand','million','billion','trillion','quadrillion','quintillion','sextil
lion','septillion','octillion','nonillion'];

  var SCALE_WORD_SOUTH_ASIAN =
['','thousand','lakh','crore','arab','kharab','neel','padma','shankh','***
','***'];


  var GROUP_SIZE = (typeof IS_SOUTH_ASIAN != "undefined" &&
IS_SOUTH_ASIAN) ? 2 : 3;

  var SCALE_WORD = (typeof IS_SOUTH_ASIAN != "undefined"
&& IS_SOUTH_ASIAN) ? SCALE_WORD_SOUTH_ASIAN :
SCALE_WORD_WESTERN;
```

```
 // Return string of first three digits, padded with zeros if needed
  function get_first_3(str)}
    return ('000' + str).substr؛((٣)-)
{
  function get_first(str) { //-- Return string of first GROUP_SIZE digits,
padded with zeros if needed, if group size is 2, make it size 3 by
prefixing with a '0'
    return (GROUP_SIZE == 2 ? '0' : '') + ('000' + str).substr(-
(GROUP_SIZE))؛
{



 // Return string of digits with first three digits chopped off
  function get_rest_3(str)}
    return str.substr(0, str.length - 3)؛
{
  function get_rest(str) { // Return string of digits with first
GROUP_SIZE digits chopped off
    return str.substr(0, str.length - GROUP_SIZE)؛
{


 // Return string of triplet convereted to words
  function triplet_to_words(_3rd, _2nd, _1st)}
    return  (_3rd == '0' ? '' : ONES_WORD[_3rd] + ' hundred ') +
'_)          st == '0' ? TENS_WORD[_2nd] : TENS_WORD[_2nd] &&
TENS_WORD[_2nd] + ('' // '-' +
          (ONES_WORD[_2nd + _1st] || ONES_WORD[_1st]);  //-- 1st
one returns one-nineteen - second one returns one-nine
```

```
{

 // Add to result, triplet words with scale word
  function add_to_result(result, triplet_words, scale_word)}
    return triplet_words ? triplet_words + (scale_word && ' ' +
scale_word || '') + ' ' + result : result;
{

  function recurse (result, scaleIdx, first, rest)}
    if (first == '000' && rest.length === 0) return result;
    var newResult = add_to_result (result, triplet_to_words (first[0],
first[1], first[2]), SCALE_WORD[scaleIdx]);
    return recurse (newResult, ++scaleIdx, get_first(rest), get_rest(rest));
{

  return recurse ('', 0, get_first_3(String(int)), get_rest_3(String(int)));
}
```