## 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**

***]3,2,1[ = m***

***]6,5,4[ = n***

***]9,8,7[ = 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]]7 ,8 ,9 ,4 ,5 ,6[ ,]1 ,2 ,3[[ =***

***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.

***“[ → ]3,6,4[one”, “four”, “six]”***

***“[ → ]2,2,4,2,9[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;))1(-(***

# *}*

*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 ') +*

*3\_( 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)));*

*}*