001-099

001. Length of a List

Given a list **L**, return length of it.

Example 1:
Input: L = [1, 2, 3, 4, 5, 6, 7]
Output: 7

Example 2:
Input: L = []
Output: 0

002. Reverse a List

Given a list **L**, return a reversed list.

Example 1:
Input: L = [1, 2, 3, 4, 5, 6, 7]
Output: [7, 6, 5, 4, 3, 2, 1]

Example 2:
Input: L = []
Output: []

Example 3:
Input: L = [element]
Output: [element]

003. Maximum Value

Given a number **A** and a number **B**, return a maximum value.

Example 1:
Input: A = 10, B = 3
Output: 10

Example 2:
Input: A = 1, B = 7
Output: 7

Example 3:
Input: A = 2, B = 2
Output: 2

004. Maximum Value in a List

Given a list **L** of numbers, return a maximum value.

Example 1:
Input: L = [1, 7, 2, -3, 5, 0]
Output: 7

Example 2:
Input: L = [4]
Output: 4

Example 3:
Input: L = [-1, -9, -4]
Output: -1

Constraints:
- 1 <= Length of L

005. Membership

Given an element **X** and a list **L**, return true if **X** is a member of **L**, false otherwise.

Example 1:
Input: X = alex, L = [bob, james, alan, alex, simon]
Output: true

Example 2:
Input: X = sam, L = [bob, james, alan, alex, simon]
Output: false

Example 3:
Input: X = 5, L = [1, 2, 3, 4, 5]
Output: true

Example 4:
Input: X = 0, L = [1, 2, 3, 4, 5]
Output: false

Example 5:
Input: X = 0, L = []
Output: false

006. Parity

Given an integer **N**, return atom **even** if the parity of **N** even, otherwise return atom **odd**.

Example 1:
Input: N = 5
Output: odd

Example 2:
Input: N = 8
Output: even

007. List Length Parity

Given a list **L**, return atom **even** if the list's length parity is even, otherwise return atom **odd**.

Example 1:
Input: L = [1, 2, 3, 4, 5, 6, 7]
Output: odd

Example 2:
Input: L = [1, 2, 3, 4]
Output: even

008. Checking List Length Parity

Given a list **L**. Define two functions: **even_length** and **odd_length,** so that they return are true if their argument is a list of even or odd length respectively.

Example 1:
Input: L = [1, 2, 3, 4, 5, 6, 7]
Call: even_length(L)
Output: false
Call: odd_length(L)
Output: true

Example 2:
Input: L = [1, 2, 3, 4]
Call: even_length(L)
Output: true
Call: odd_length(L)
Output: false

009. Sum of Elements in a List

Given a list **L** of numbers, return the sum of all elements in the list.

Example 1:
Input: L = [1, 2, 3, 4, 5, 6, 7]

Output: 28

Example 2:
Input: L = []
Output: 0

Example 3:
Input: L = [12]
Output: 12

Example 4:
Input: L = [10, 0, -5]
Output: 5

010. Removing Last 3 Elements in a List

Given a list **L**, return a list without 3 last elements.

Example 1:
Input: L = [1, 2, 3, 4, 5, 6, 7]
Output: [1, 2, 3, 4]

Example 2:
Input: L = []
Output: 0

Example 3:
Input: L = [sun, moon]
Output: []

Example 4:
Input: L = [jane, laura, jerry, katty]
Output: [jane]

011. Last Element

Given a list **L**, return the last element.

Example 1:
Input: L = [1, 2, 3, 4, 5, 6, 7]
Output: 7

Example 2:
Input: L = [sun, moon]
Output: moon

Example 3:
Input: L = [1]

Output: 1

Example 4:
Input: L = [jane, laura, jerry, katty]
Output: katty

Constraints:

- 1 <= Length of L

012. Deleting an Item

Given an item **X** and a list **L**, return a list in which the first occurrence of item **X** has been removed**.**

Example 1:
Input: X = 2, L = [1, 2, 3, 4, 5, 6, 7]
Output: [1, 3, 4, 5, 6, 7]

Example 2:
Input: X = elisa, L = [bob, mark, elisa, greg]
Output: [bob, mark, greg]

Example 3:
Input: X = 1, L = [1]
Output: []

013. Ordered List

Given a list **L** of numbers, return **true** if the list is ordered, **false** otherwise**.**

Example 1:
Input: L = [1, 2, 3, 4, 5, 6, 7]
Output: true

Example 2:
Input: L = [1, 2, 7, 5, 9]
Output: false

Example 3:
Input: L = [10]
Output: true

Constraints:

- 1 <= Length of L

014. Shift a List

Given a list **L**, return a list ‘shifted rotationally‘ by one element to the left**.**

Example 1:
Input: [1, 2, 3, 4, 5, 6, 7]
Output: [2, 3, 4, 5, 6, 7, 1]

Example 2:
Input: [1, 2, 7, 5, 9]
Output: [9, 1, 2, 7, 5]

Example 3:
Input: [sun]
Output: [sun]

Example 4:
Input: [ben, julia, antony]
Output: [antony, ben, julia]

015. Translate digits to words

Given a list **L** of numbers between 0 and 9, translate to a list of the corresponding words**.**

Example 1:
Input: [1, 2, 3, 4]
Output: [one, two, three, four]

Example 2:
Input: [7, 5, 9]
Output: [seven, five, nine]

Example 3:
Input: [6]
Output: [six]

016. Between

Given two integer numbers **N1**, **N2**, return the ordered list of all integers between **N1** and **N2**, **N1** <= N < **N2.**

Example 1:
Input: N1 = 2, N2 = 7
Output: [2, 3, 4, 5, 6]

Example 2:

Input: N1 = 0, N2 = 3
Output: [0, 1, 2]

Example 3:
Input: N1 = 9, N2 = 4
Output: []

017. Factorial

Given an integer number **N**, return the factorial of **N.**

Example 1:
Input: 0
Output: 1

Example 2:
Input: 5
Output: 120

Example 3:
Input: 8
Output: 40320

Constraints:
- 0 <= N

018. Move Zeroes

Given a list **L** of integer numbers, move all 0's to the end of it while maintaining the relative order of the non-zero elements.

Example 1:
Input: [0,1,0,3,12]
Output: [1,3,12,0,0]

Example 2:
Input: [0]
Output: [0]

Example 3:
Input: [0,2,0,0,5,6,0,5]
Output: [2,5,6,5,0,0,0,0]

019. Fibonacci Sequence

Given an integer number **N**, return the list of Fibonacci sequence, up until the Nth term**.**

Example 1:
Input: 1

Output: [0, 1]

Example 2:
Input: 5
Output: [0,1,1,2,3,5]

Example 3:
Input: 8
Output: [0,1,1,2,3,5,8,13,21]

Example 4:
Input: 11
Output: [0,1,1,2,3,5,8,13,21,34,55,89]

Constraints:

- 0 <= N

020. Divide a List

Given a list **L**, return two lists **L1** and **L2**, so that the elements of **L** are partitioned between **L1** and **L2**, and **L1** and **L2** are of approximately the same length**.**

Example 1:
Input: L = [0, 1]
Output: L1 = [0], L2=[1]

Example 2:
Input: L = [a, b, c, d, e]
Output: L1=[a, c, e], L2=[b, d]

Example 3:
Input: L=[1, 2, 3, 4]
Output: L1=[1, 3], L2=[2, 4]

021. Flatten a List

Given a list **L,** where **L** can be a list of lists, return a list "flattened" so that the elements of List's sublists are reorganized as one plain list**.**

Example 1:
Input: [0, [1, 2, 3], 4, [5, 6, [7, 8]], 9]
Output: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

Example 2:
Input: [[[a, b]]]
Output: [a, b]

Example 3:

Input: [[[[]]]]
Output: []

022. Permutations

Given a list **L,** return all permutations of the list **L.**

Example 1:
Input: [1, 2]
Output: [1, 2], [2, 1]

Example 2:
Input: [1]
Output: [1]

Example 3:
Input: [a, b, c]
Output: [a, b, c], [a, c, b], [b, a, c], [b, c, a], [c, a, b], [c, b, a]

023. Sublist

Given a list **S** and a list **L,** return **true** if **S** is a sublist of **L**, **false** otherwise**.**

Example 1:
Input: S = [1], L = [1,2,3]
Output: true

Example 2:
Input: S = [b, c], L = [a, b, b, c, d]
Output: true

Example 3:
Input: S = [a, b, c], L = [a, b, d, c, e]
Output: false

024. Subset

Given a list **S** and a list **L,** return **true** if **S** is a subset of **L**, **false** otherwise**.**

Example 1:
Input: S = [1], L = [1, 2, 3]
Output: true

Example 2:
Input: S = [b, c], L = [a, b, b, d, c]
Output: true

Example 3:
Input: S = [a, b, f], L = [a, b, d, c, e]
Output: false

025. Split a List of Numbers into Positive and Negative ones

Given a list **L** of numbers**,** split **L** into two lists: positive ones (including zero) and negative ones.

Example 1:
Input: L = [1, 2, 3]
Output: P = [1, 2, 3], N = []

Example 2:
Input: L = [0, -1, 2, -3, -4]
Output: P = [0, 2], N = [-1, -3, -4]

Example 3:
Input: L = [1, -1]
Output: P = [1], N = [-1]

026. Split a Mixed List into Atoms List and Numbers List

Given a mixed list **L** of numbers and atoms**,** split **L** into two lists: atoms only list and numbers only list.

Example 1:
Input: L = [1, one, 2, two, 3]
Output: **A** = [one, two], N = [1, 2, 3]

Example 2:
Input: L = [0, hello, -1, 2, -3, world]
Output: A = [hello, world], N = [0, -1, 2, -3]

Example 3:
Input: L = []
Output: A = [], N = []

027. Doubled Numbers

Given a list **L** of numbers**,** return a list with doubled numbers.

Example 1:
Input: [1, 2, 3]
Output: [2, 4, 6]

Example 2:
Input: [0, 50, 100]

Output: [0, 100, 200]

Example 3:
Input: []
Output: []

028. Mean value

Given a list **L** of numbers**,** return the mean value of the list.

Example 1:
Input: [1, 2, 3, 4, 5]
Output: 3

Example 2:
Input: [100, 200]
Output: 150

Example 3:
Input: [7]
Output: 7

Constraints:

- 1 <= Length of L

029. Median

Given a list **L** of numbers**,** return the median of the list.

Example 1:
Input: [-1, 1, 2, 3, 40]
Output: 2

Example 2:
Input: [23, 45, 67, 1, 4, 120, -3]
Output: 23

Example 3:
Input: [1, 2]
Output: 1.5

Constraints:

- `1 <= Length of L`

030. Nth Element

Given a list **L,** return **N**th element of the list (zero based).

Example 1:
Input: L = [1, 2, 3, 4, 5, 6], N = 2
Output: 3

Example 2:
Input: L = [mark, john, leo, george], N = 3
Output: george

Example 3:
Input: L = [1], N = 0
Output: 1

Constraints:

- 1 <= Length of L
- N >= 0, N < Length of L

031. Bubble Sort

Given a list **L,** return the sorted list (use Bubble Sort).

Example 1:
Input: [1, 4, 2, 7, 6, 3, 5]
Output: [1, 2, 3, 4, 5, 6, 7]

Example 2:
Input: [3, 2, 1]
Output: [1, 2, 3]

032. Merged Sorted Lists

Given sorted lists **L1, L2,** return the merged sorted list.

Example 1:
Input: L1 = [1, 3, 5], L2 = [2, 4, 6]
Output: [1, 2, 3, 4, 5, 6].

Example 2:
Input: L1 = [-1, 1], L2 = [0]
Output: [-1, 0, 1]

Example 3:
Input: L1 = [2, 7, 15, 35, 60, 115], L2 = [40, 100]
Output: [2, 7, 15, 35, 40, 60, 100, 115]

033. Matrix with Random Numbers

Given an integer number **N**, return the matrix **N**x**N** with random integer numbers in range 0..9.

Example 1:
Input: N = 2
Output: [[1, 6], [8, 7].

Example 2:
Input: N = 3
Output: [[1, 5, 8], [3, 2, 8], [7, 0, 6])

034. Quick Sort

Given a list **L,** return the sorted list (use Quick Sort).

Example 1:
Input: [3, 2, 1]
Output: [1, 2, 3]

Example 2:
Input: [2, 3, 1]
Output: [1, 2, 3]

Example 3:
Input: [1, 2, 3]
Output: [1, 2, 3]

035. Modified Quick Sort

Given a list **L,** return the sorted list (use Modified Quick Sort).

Example 1:
Input: [3, 2, 1]
Output: [1, 2, 3]

Example 2:
Input: [2, 3, 1]
Output: [1, 2, 3]

Example 3:
Input: [1, 2, 3]
Output: [1, 2, 3]

036. Insertion Sort

Given a list **L,** return the sorted list (use Insertion Sort).

Example 1:
Input: [3, 2, 1]
Output: [1, 2, 3]

Example 2:
Input: [2, 3, 1]
Output: [1, 2, 3]

Example 3:
Input: [1, 2, 3]
Output: [1, 2, 3]

037. Palindrome

Given a list **L,** return true if list L is a palindrome, otherwise false.

Example 1:
Input: [m, a, d, a, m]
Output: true

Example 2:
Input: [c, a, t]
Output: false

Example 3:
Input: [m, a, m]
Output: true

038. Sub sum

Given a list **L** of positive integer numbers and a sub sum **S,** return the subset of these numbers where sum of subset equal sub sum **S**, otherwise an empty list.

Example 1:
Input: L = [1, 2, 3, 4], S = 5
Output: [1, 4]

Example 2:
Input: L = [1, 2, 3, 4], S = 3
Output: [1, 2]

Example 3:
Input: L = [1, 2, 3, 4], S = 20
Output: []

039. Contains Duplicate (LC217)

Given a list L of numbers, return true if any value appears **at least twice** in the array, and return false if every element is distinct.

Example 1:
Input: L = [1,2,3,1]
Output: true

Example 2:
Input: L = [1,2,3,4]
Output: false

Example 3:
Input: L = [1,1,1,3,3,4,3,2,4,2]
Output: true

040. Parent Relations

Define parent/child relations using records, create tiny db of parents, print db.

Example 1:
Output: John is parent of Ann.
Julia is parent of Mike.
...

041. Parent Relations (ETS)

Define parent/child relations using records and ets, create tiny db of parents, print db.

Example 1:
Output: John is parent of Ann.
Julia is parent of Mike.
…

042. Parent Relations (DETS)

Define parent/child relations using records and dets, create tiny db of parents, print db.

Example 1:
Output: John is parent of Ann.
Julia is parent of Mike.
…

043. Parent Relations (MNESIA)

Define parent/child relations using records and mnesia, create tiny db of parents, print db.

Example 1:
Output: John is parent of Ann.

Julia is parent of Mike.

044. Heap Sort

Given a list **L** of numbers**,** return the sorted list (use Heap Sort).

Example 1:
Input: [3, 2, 1]
Output: [1, 2, 3]

Example 2:
Input: [2, 3, 1]
Output: [1, 2, 3]

Example 3:
Input: [1, 2, 3]
Output: [1, 2, 3]

045. Odd-Even Sort

Given a list **L** of numbers**,** return the sorted list (use Odd-Even Sort).

Example 1:
Input: [3, 2, 1]
Output: [1, 2, 3]

Example 2:
Input: [2, 3, 1]
Output: [1, 2, 3]

Example 3:
Input: [1, 2, 3]
Output: [1, 2, 3]

046. Counting Sort

Given a list **L** of integer numbers**,** return the sorted list (use Counting Sort).

Example 1:
Input: [3, 2, 1]
Output: [1, 2, 3]

Example 2:
Input: [2, 3, 1]
Output: [1, 2, 3]

Example 3:
Input: [1, 2, 3]

Output: [1, 2, 3]

Constraints:

- 0 <= Number < 100

047. Pascal's Triangle

Given an integer number **N,** return Pascal's Triangle of **N**.

Example 1:
Input: 0
Output: [1]

Example 2:
Input: 2
Output: [1], [1, 1]

Example 3:
Input: 4
Output: [1] [1, 1], [1, 2, 1], [1, 3, 3, 1], [1, 4, 6, 4, 1]

Constraints:

- 0 <= N

048. Timer. Counter

Create a counter from 0 to 9 that will increment every second.

049. Timer. Counter. Notify

Create a counter from 0 to 9 that will increment every second. Notify when done.

050. Timer. Counter. Notify 2.

Create a counter from 0 to 9 that will increment every second. Notify to print counter. Notify when done.