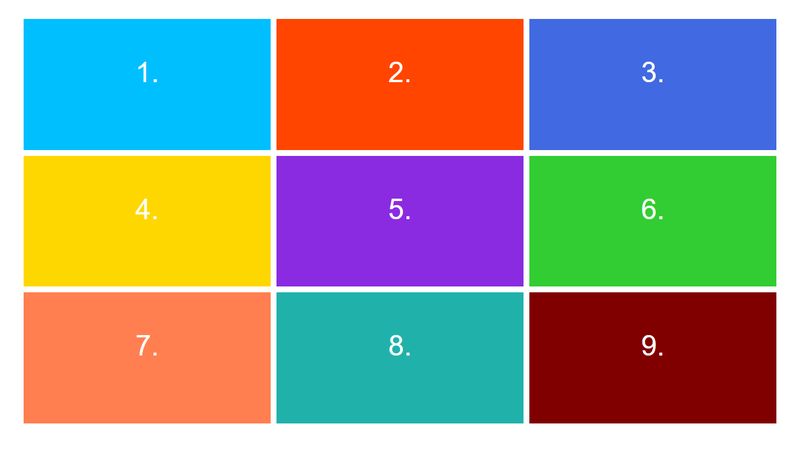
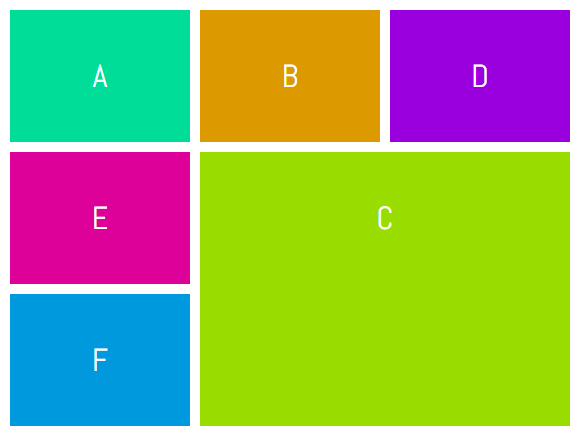
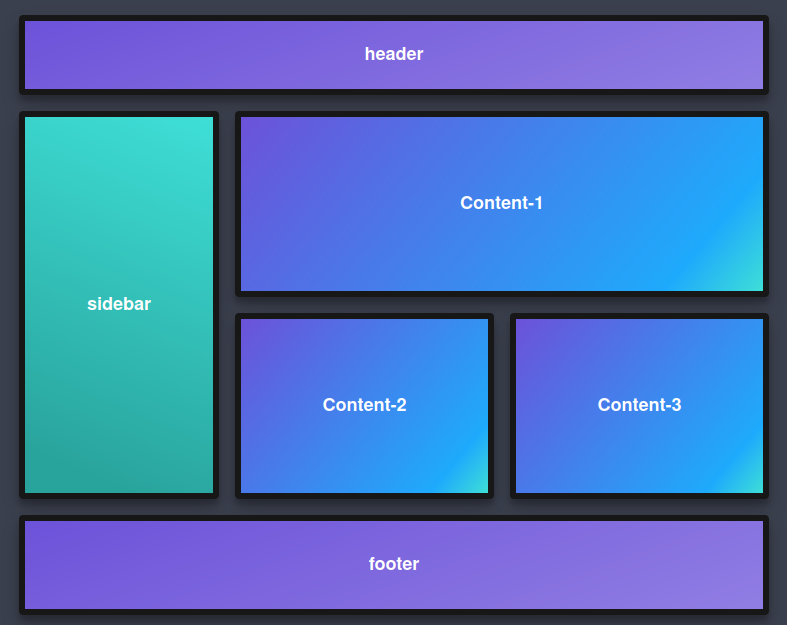
Total=7

Must learn the basics of CSS Flex before attempting the following question.

**Q1: Learn flex in CSS and create following 3 demos in single project with HTML+CSS.**







Level: Easy

Do them in plain JavaScript.

**Q2: Learn and practice following concepts and practice the following concepts.**

1. Object
2. Array
3. Functions

**Q3: Find the Smallest and Biggest Numbers.**

Create a function that takes an array of numbers and return both the minimum and maximum numbers, in that order.

### Examples

minMax([1, 2, 3, 4, 5]) ➞ [1, 5]

minMax([2334454, 5]) ➞ [5, 2334454]

minMax([1]) ➞ [1, 1]

**Q4: Drink Sorting.**

You will be given an array of drinks, with each drink being an object with two properties: name and price. Create a function that has the drinks array as an argument and return the drinks objects sorted by price in ascending order.

Assume that the following array of drink objects needs to be sorted:

drinks = [

{name: "lemonade", price: 50},

{name: "lime", price: 10}

]

The output of the sorted drinks object will be:

### Examples

sortDrinkByPrice(drinks) ➞ [{name: "lime", price: 10}, {name: "lemonade", price: 50}]

**Q5: Older Than Me.**

Create a method in the Person class which returns how another person's age compares. Given the instances p1, p2 and p3, which will be initialised with the attributes name and age, return a sentence in the following format:

**{other person name} is {older than / younger than / the same age as} me.**

### Examples

p1 = Person("Samuel", 24)

p2 = Person("Joel", 36)

p3 = Person("Lily", 24)

p1.compareAge(p2) ➞ "Joel is older than me."

p2.compareAge(p1) ➞ "Samuel is younger than me."

p1.compareAge(p3) ➞ "Lily is the same age as me."

### Notes

* You can also solve it by creating a function compareAge and pass both person object in it and show proper message.

**Q6: Tuck in Array.**

Create a function that takes two arrays and insert the second array in the middle of the first array.

### Examples

tuckIn([1, 10], [2, 3, 4, 5, 6, 7, 8, 9]) ➞ [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

tuckIn([15,150], [45, 75, 35]) ➞ [15, 45, 75, 35, 150]

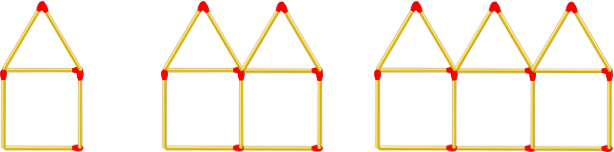
tuckIn([[1, 2], [5, 6]], [[3, 4]]) ➞ [[1, 2], [3, 4], [5, 6]]

### Notes

* The first array always has two elements.
* Use the **spread syntax** to solve this challenge.

**Q7: Matchstick Houses.**

This challenge will help you interpret mathematical relationships both algebraically and geometrically.



Create a function that takes a number (step) as an argument and returns the number of matchsticks in that step. See step 1, 2 and 3 in the image above.

### Examples

matchHouses(1) ➞ 6

matchHouses(4) ➞ 21

matchHouses(87) ➞ 436

### Notes

* Step 0 returns 0 matchsticks.
* The input (step) will always be a non-negative integer.
* Think of the input (step) as the total number of houses that have been connected together.