用 Javascript 解决下面的问题

1. 一个人投篮 12 次，命中 8 球；假如命中为 1，不命中为 0，那么一次投篮的组合可以表示如 000011111111 所示。请问所有可能的排列组合中，出现连续命中 4 球及以上的比例有多大？（提示：可以穷举所有排列，然后找出其中含有 1111 的排列）

2. 黄小明是老司机了，他每天任务就是送一群人去上班。但是每天送的人数不一样，这些人上车的时间也不一样。小明总是要等到人齐才能开车。
完成函数 `driveCustomers`，它接受不定数量的参数，这是参数都是函数，每个函数代表一个人。**这些函数都接受一个回调函数作为参数，当回调函数被调用的时候说明这个人已经上车了**，回调函数会被传入人名。例如：

```
const MissLi = (callback) => {

  setTimeout(() => {

    callback('MissLi')

  }, 10) // 上车时间不一定

}



const MrWang = (callback) => {

  setTimeout(() => {

    callback('MrWang')

  }, 3) // 上车时间不一定

}



driveCustomers(MissLi, MarWang, ...)
```

请你完成 `driveCustomers` 函数，它的作用是：当人都到齐以后，按上车的时间顺序把人名放到一个数组里面然后传给 `drive` 函数，正式开车。例如：`drive(['MrWang', 'MissLi'])`。

你只需要完成 `driveCustomers` 函数，`drive` 函数已经可以直接使用。

3. 在函数式编程当中有一个很重要的概念就是函数组合，实际上就是把处理数据的函数像管道一样连接起来，然后让数据穿过管道得到最终的结果。例如：

```
const add1 = (x) => x + 1

const mul3 = (x) => x * 3

const div2 = (x) => x / 2



div2(mul3(add1(add1(0)))) // => 3
```

而这样的写法可读性明显太差了。我们可以构建一个 `compose` 函数，它接受任意多个函数作为参数（这些函数都只接受一个参数），然后 `compose` 返回的也是一个函数，达到以下的效果：

```
const operate = compose(div2, mul3, add1, add1)

operate(0) // => 相当于 div2(mul3(add1(add1(0))))

operate(2) // => 相当于 div2(mul3(add1(add1(2))))
```

简而言之：`compose` 可以把类似于 `f(g(h(x)))` 这种写法简化成 `compose(f, g, h)(x)`。请你完成 `compose` 函数的编写。

额外挑战：你能通过 1~2 行代码实现 `compose` 吗。

4，树状结构是一种常见的数据结构也是一种常见的前端展示元素，现提出如下需求：

需求 1：根据给出的列表数据，生成相应的树状结构，树状结构的展现形式不限。列表树、图形树皆可。（数据可参考样例数据）

需求 2：单击树种的某个节点，按深度优先的顺序打印出该节点及每个子节点的 id，如： 样例数据生成的树中点击 a， 打印； a-> b -> c -> d -> e -> f -> g -> h -> i -> j -> k -> l -> m

需求 3：鼠标移入某节点时，按宽度优先的顺序打印出该节点及每个子节点的 id；如： 样例数据生成的树中点击 a， 打印；a-> b -> d -> h -> i -> e-> j -> c -> f -> k -> g -> l -> m

注意： 同一层的顺序不必限定，以上两种情况下 比如 a -> b -> c 和 a -> c -> b 这两种顺序是等价的 （本题目不限前端框架，但不允许使用第三方树组件包，如 antd 的 Tree 组件）

ID	父 ID
a	null
b	a
c	a
d	b
e	b
f	c
g	c
h	d
i	d
j	e
k	f
l	g
m	g

题 4 样例数据