**Функциональное программирование**

**Пример**

Для начала разберем простую задачу с собеседования в Amazon “Поиск знаменитости”.

**Условие задачи:**

Дан двумерный массив, в котором каждая строка представляет собой набор из нулей и единиц и обозначает одного человека из группы. Рассмотрим для примера строку [1, 0, 1, 0, 1, 1]. Человек под индексом 0 знает человека под индексом 2, поэтому там стоит 1, но не знает человека под индексом 1, поэтому там стоит 0. Логично, что каждый человек знает самого себя, поэтому элемент с индексом равным индексу человека всегда равен 1. Требуется найти знаменитость. Знаменитостью является человек, которого знают все, и который никого не знает.

Например, группа из 3 людей:

[1, 0, 1] – нулевой знает второго, но не знает первого.

[1, 1, 1] – первый знает и нулевого, и второго.

[0, 0, 1] – второй никого не знает.

В этом случае второй человек будет являться знаменитостью. Потому что его все знают, он никого не знает.

**Решение задачи:**

Первая идея, которая приходит в голову, брать строку под номером i и проверять, равен ли какой-нибудь элемент (в этой строке), кроме i-ого, 1. Если нет, то проверять , состоит ли i-ый столбец только из единиц. Таким образом, в худшем случае, нам нужно будет пройтись по массиву (длинной n) n^2 раз.

Но, проанализировав условия задачи, можно сделать вывод, что достаточно проверять людей попарно. Например, возьмем пару 0 и 2 из примера в условии. Нулевой знает второго, а так как знаменитость знает только самого себя, этот вариант можно отсечь. Второй не знает нулевого, поэтому пока он нам подходит. Сравним пару 1 и 2. Аналогичная ситуация, первый нам не подходит. Из вариантов остался только второй. Его нам и следует проверить на “знаменитость”.

Таким методом решения нам придется пройти по массиву всего 3n раз.

**Реализация задачи на языке JS в функциональном стиле:**

Реализуем попарный поиск претендента на знаменитость.

Создадим функцию search. Как аргументы она будет принимать двумерный массив, счетчик начала поиска (по умолчанию 0) и счетчик конца поиска (по умолчанию размер массива -1). Возвращать функция будет номер претендента на знаменитость.

Search, пока счетчики не станут равны, будет рекурсивно вызываться со следующей парой. Следующая пара зависит от результата сравнения предыдущей.

*const search* = (people, l = 0, r = people.length - 1) => {  
 *if* (l === r) *return* l;  
 *switch* (*checkPair*(people, l, r)) {  
 *case* 0:  
 *return search*(people, ++l, --r);  
 *case* 1:  
 *return search*(people, ++l, r);  
 *case* -1:  
 *return search*(people, l, --r);  
 }  
}

Нужно создать функцию checkPair, которая будет принимать массив и индексы людей для анализа (i и j) как аргументы и возвращать 1, 0, -1.

* 1 – если i знает j.
* 0 – если i не знает j и j не знает i (в таком случае оба человека не подходят).
* -1 – если j знает i.

const checkPair = (people, i, j) => {

// Наглядная реализация  
 // if (people[i][j] === people[j][i]) return 0;  
 // if (people[i][j] === 1) return 1;  
 // if (people[j][i] === 1) return -1;

// Краткая реализация  
 return people[i][j] - people[j][i];  
}

Для того, чтобы проверить претендента на истинную знаменитость, создадим функцию checkFamous. Аргументами будут массив и индекс претендента, возвращаемым значением true/false.

Достаточно проверить три условия:

1. чтобы в строке была только одна единица.
2. чтобы эта единица имела индекс строки.
3. чтобы столбец с индексом строки состоял только из единиц. Для проверки столбца создадим рекурсивную функцию checkColumn.

*const checkColumn* = (people, i, j) => {  
 *if* (i === people.length) *return true*;  
  
 *if* (people[i][j] === 0) *return false*;  
 *else  
 return checkColumn*(people, ++i, j);  
}

*const checkFamous* = (people, i) => {  
 *return* people[i].filter(e => e !== 1).length === people.length - 1 && *checkColumn*(people, 0, i) && people[i][i] === 1;  
}

Осталось связать функции search и checkFamous. Создадим функцию searchFamous, которая, используя написанные нами функции, будет решать главную задачу -> искать знаменитость.

*const searchFamous* = (people) => {  
 *const* i = *search*(people);  
 *if* (*checkFamous*(people, i))  
 *return* i;  
 *else  
 return* -1;  
}

**Полный код:**

*const* people = () => [  
 [1, 0, 1, 0, 0, 0],  
 [0, 1, 1, 0, 0, 0],  
 [0, 0, 1, 0, 0, 0],  
 [0, 0, 1, 1, 0, 0],  
 [0, 0, 1, 0, 1, 0],  
 [0, 0, 1, 0, 0, 1]  
];  
  
*const checkPair* = (people, i, j) => {  
 *return* people[i][j] - people[j][i];  
}  
  
*const checkColumn* = (people, i, j) => {  
 *if* (i === people.length) *return true*;  
  
 *if* (people[i][j] === 0) *return false*;  
 *else  
 return checkColumn*(people, ++i, j);  
}  
  
*const search* = (people, l = 0, r = people.length - 1) => {  
 *if* (l === r) *return* l;  
 *switch* (*checkPair*(people, l, r)) {  
 *case* 0:  
 *return search*(people, ++l, --r);  
 *case* 1:  
 *return search*(people, ++l, r);  
 *case* -1:  
 *return search*(people, l, --r);  
 }  
}  
  
*const checkFamous* = (people, i) => {  
 *return* people[i].filter(e => e !== 1).length === people.length - 1 && *checkColumn*(people, 0, i) && people[i][i] === 1;  
}  
  
*const searchFamous* = (people) => {  
 *const* i = *search*(people);  
 *if* (*checkFamous*(people, i))  
 *return* i;  
 *else  
 return* -1;  
}  
  
console.log(*searchFamous*(people()));

**Задачи**

Правила выполнения:

* Данный блок является контрольной точкой. Можно получить 3 вида отметок.
  + “удовлетворительно” – нужно набрать 4 балла.
  + “хорошо” – нужно набрать 5 баллов.
  + “отлично” – нужно набрать 7 баллов.
* За выполненную задачу +0.5 балла.
* За то, что решение выполнено в функциональном стиле, +0.5 балла.
* За оптимизированные версии задач 2, 4, 7 также начисляется +0,5 балла.
* Нет ограничения по количеству используемых второстепенных функций.

1. Своеобразное вычитание

Даны 2 целых числа **a** и **b**. Реализуйте функцию, которая будет принимать как аргументы **a** и **b**, и производить арифметику по логике:

Операцию нужно повторить **b** раз.

* если последняя цифра числа **a** не равна нулю, число уменьшается на единицу;
* если последняя цифра числа **a** равна нулю, число делится на 10 (то есть удаляет его последнюю цифру).

Например:

**a** = 512, **b** = 4.

512 -> 511 -> 510 -> 51 -> 50

Функция должна возвращать измененное число **a**.

1. Найти центр

Дан массив из точек на координатной прямой неизвестной размерности.

[ 1, 1 ]  
 [ 0, 2 ]  
 [ 0, 1 ]  
 [ 1, 3 ]  
 [ 1, 4 ]  
 [ 2, 1 ]  
 […, …]

Введем понятие центральной точки и соседей:

* точка (x', y') называется правым соседом для (x, y), если x' > x и y' = y
* точка (x', y') называется левым соседом для (x, y), если x' < x и y' = y
* точка (x', y') называется нижним соседом для (x, y), если x' = x и y' < y
* точка (x', y') называется верхним соседом для (x, y), если x' = x и y' > y
* точка (x, y) называется центральной, если имеет хотя бы одного соседа каждого типа (в сумме минимум 4)

Реализуйте функцию, которая будет принимать как аргумент массив точек и возвращать количество центральных точек.

1. Камушки

Дан массив, который хранит в себе информацию о попарном взвешивании камушков друг с другом. Каждый камень обозначается какой-то буквой A, B, C, D…, а результат взвешивания true/false.

Например:

[“A”, “B”, true]  
 [“C”, “B”, false]   
 [“A”, ”C”, true ]   
 [“D”, ”A”, false]   
 [“B”, ”D”, true]  
 [“D”, “C”, true]

Комментарий: *Камней может быть любое количество. Обозначение [ “A”, “B”, ? ] -> true – означает, что A весит больше B, false – то, что A меньше B.*

Реализуйте функцию, которая будет принимать как аргумент массив c информацией о взвешиваниях и возвращать массив только из камней, отсортированных по весу.

Пример возвращаемого значения: [“A”, “D”, “B”, “C”].

1. Ребята

Ребята решили поиграть в игру «Снежный ком». Правила игры следующие: сначала все становятся в ряд, затем первый ребенок произносит своё имя. После этого второй ребенок произносит имя первого, а затем произносит своё имя. Затем третий ребенок произносит имя первого, затем второго, а затем своё.

Дан массив из имен ребят (любой размерности), собравшихся во дворе, и число k.

Реализуйте функцию, которая будет принимать массив с именами детей и число k как аргументы и возвращать имя, которое будет стоять в последовательности произносимых имен на позиции k.

Например:

* Names = [“Миша”, ”Гена”, ”Маша”].
* k = 4.

Последовательность будет такой: Миша -> Миша -> Гена -> Миша -> Гена -> Маша. На позиции 4 стоит имя Миша, это и будет нашим ответом.

1. Вова и Шахматы

Вова захотел поиграть в шахматы. Фигуры он решил сделать из бумаги, а доску из листа картона. Фигуры Вова сделал сам, а доску попросил сделать очень невнимательного товарища. Помогите Вове проверить доску на правильность.

Дан массив (length = 8) из строк, каждая строка (длинною 8 символов) состоит только из букв W (white) и B(black).

Например:  
 [“WBWBWBWB”]  
 [“BWBWBWBW”]  
 [“WBWBWBWB”]   
 [“BWBWBWBW”]  
 [“WBWBWBWB”]  
 [“BWBWBWBW”]  
 [“WBWBWBWB”]  
 [“BWBWBWBW”]

Известно, что в правильной шахматной доске нет соседних по сторонам клеток с одинаковым цветом, а левая верхняя клетка — белого цвета. Друзья могут выбрать любую строку имеющейся доски и циклически сдвинуть клетки выбранной строки, то есть последнюю клетку (самую правую) поставить на первое место в строке, а остальные сдвинуть на одну позицию вправо. Описанную операцию можно выполнять несколько раз (можно вообще не выполнять).

Реализуйте функцию, которая будет принимать как аргумент массив, обозначающий доску, и возвращать true или false в зависимости от того, можно ли превратить доску в правильную шахматную.

1. Сортировка на чётность

Дан массив из целых положительных чисел. Длина массива не ограничена.

Реализуйте функцию, которая будет сортировать данный массив по закону [нечетное, четное, нечетное, четное…]. Величина числа никак не влияет на сортировку. Функция должна возвращать отсортированный массив или пустой массив, если сортировка невозможна.

Например, дан массив [2, 3, 2, 5, 10, 1, 1, 6]. Один из вариантов его сортировки будет выглядеть как: [1, 2, 3, 2, 5, 10, 1, 6].

1. Поймать Гроулита

Миша решил, что станет великим мастером покемонов. Его любимый – Гроулит. Но пока он слишком мал, чтобы ловить настоящих покемонов, поэтому довольствуется поиском его имени в строках.

Дана строка из букв. Например: “ГроуГрлрайиоузилтт”.

Реализуйте функцию, которая будет как аргумент получать строку, а возвращать максимальное количество слов “Гроулит”, которое можно составить из этих букв.

*Комментарий: из примера выше Миша может составить только 2 слова. Важно учесть, что слово всегда должно начинаться с заглавной буквы Г.*

1. Чат

Для сервера простого мессенджера понадобилась программа, которая отслеживает трафик по отправке сообщений.

Вам дан массив неизвестной размерности, в котором каждый элемент обозначает одно действие в чате:

* +Имя – подключение к чату.
* -Имя – отключения от чата.
* Имя:сообщение – отправка сообщения пользователем.

В подсчет трафика входит только размер текста (в символах) отправленных сообщений, умноженных на количество пользователей, которым было отправлено сообщение.

Например:

[

“+Миша”,

“+Саша”,

“-Саша”,

“+Дима”,

“+Олег”,

“Олег:Всем привет!”

]

Сообщение Олега отправится Мише и Диме. Длина сообщения – 12 символов. Трафик равен 12\*2 => 24.

*Комментарий: пробелы и любые знаки препинания тоже считаются как символы.*

Реализуйте функцию, которая как аргумент будет принимать массив с действиями и возвращать число, обозначающее количество трафика на сервере.

**Данные для тестирования (Могут быть неверными)**

1. Своеобразное вычитание

* Данные:
  + a = 90000400001240.
  + b = 24.
* Ответ: 9

1. Найти центр

* Данные:
  + Array = [

[0, 0], [0, 1], [0, -1], [1, 0],

[-1, 0], [4, 5], [4, 4], [3, 4],

[4, 3], [3, 3], [11, 2], [5, 5],

[6, 9], [3, 6], [1, 4], [9, 1],

[7, 10], [4, -1], [-3, 0], [-5, 2],

[0, 2], [0, 2], [3, 2],

[-2, 0], [4, 5], [4, 4],

[4, 6], [5, 3], [3, -1], [3, 0],

[6, 9], [2, 6], [3, -7], [6, 7],

[7, 10], [4, -1], [-5, 2], [-1, 2],

]

* Ответ: 5

1. Камушки

* Данные:
  + Array = [

["A", "B", true], ["A", "C", true], ["B", "C", true],

["A", "D", true], ["B", "D", true], ["D", "C", true],

["E", "D", true], ["E", "B", true], ["A", "E", false],

["C", "E", false]

]

* + Ответ: [“E”, ”A”, ”B”, ”D”, ”C”]

1. Ребята

* Данные:
  + Array = ["Миша", "Гена", "Маша", "Олег", "Петя", "Вова", "Вера", "Лена", "Света"].
  + k = 30.
* Ответ: Гена

1. Вова и шахматы

* Данные:
  + Array = [

["WBWBWBWB"],

["BWBWBWBW"],

["BWBWBWBW"],

["BWWBWBWW"],

["WBWBWBWB"],

["WBWBWBWB"],

["WBWBWBWB"],

["WBWBWBWB"],

]

* Ответ: false
* Данные:
  + Array = [

["WBWBWBWB"],

["WBWBWBWB"],

["WBWBWBWB"],

["WBWBWBWB"],

["WBWBWBWB"],

["WBWBWBWB"],

["WBWBWBWB"],

["WBWBWBWB"],

]

* Ответ: true

1. Сортировка на чётность

* Данные:
  + Array = [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2]
* Ответ: [1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2]

1. Поймать Гроулита

* Данные:
  + str = "ГозулиагртГтГроаугитглтГроауГрулзиораороуизлтГроуулГиатзлилит".
* Ответ: 7

1. Чат

* Данные:
  + Array = [

["+Миша"], ["+Олег"],["+Антон"]

["+Таня"], ["-Миша"],

["Таня:Как дела?"],

["Олег:Все хорошо!"],

["-Таня"], ["+Толя"],

["Толя:Я что-то пропустил?"]

]

* Ответ: 78