**Собрать информацию и не повторяться**

Дано дерево, узлы которого могут иметь некоторый набор свойств. Информация о свойствах хранится в секции info соответствующего узла.

Эти свойства могут наследоваться или переопределяться ниже по дереву. Если родительский узел имеет некоторое свойство, дочерний узел наследует его, кроме случаев, когда оно было переопределено.

Кроме того, если у родительского узла явно не указано свойство, которое имеется у всех его дочерних узлов, считается, что родительский узел также обладает этим свойством.

**Формат ввода**

Описание свойств в узлах дерева выглядит следующим образом:

{

"info": {

"oreType": "gold",

"weather": "rainy"

},

"children": [

{

"info": {

"oreType": "gold",

"weather": "rainy"

},

"children": [

{

"info": {

"oreType": "gold",

"weather": "rainy"

},

"children": []

},

{

"info": {

"oreType": "cooper",

"weather": "sunny"

},

"children": []

}

]

}

]

}

Формат:

interface InputNode {

info: Record<string, string>

children: InputNode[]

}

**Формат вывода**

Результатом работы алгоритма должен быть объект с двумя свойствами: atomicInfo и tree.

В значении свойства atomicInfo ожидается лексикографически отсортированное перечисление атомарной информации обо всех уникальных сочетаниях имени свойства/его значения. Эти уникальные сочетания следует собрать из секций info узлов дерева и сформировать из них имена характеристик.

В значении свойства tree результирующего объекта ожидается оптимизированное дерево. Оптимизация заключается в следующем:

* Если в узле явно описаны свойства, которые он и так наследует от узла выше по дереву, эти свойства следует удалить;
* Если в родительском узле отсутствует свойство, которое при этом имеется у всех его дочерних узлов, необходимо присвоить родителю это свойство, а из дочерних узлов - удалить;
* Секции info следует превратить в массивы с именами характеристик, построенных из оставшихся свойств, и также сортировать лексикографически. Имена характеристик строятся так же, как и для свойства atomicInfo.

Пример результирующего объекта:

{

"atomicInfo": [

"info\_oreType\_cooper",

"info\_oreType\_gold",

"info\_weather\_rainy",

"info\_weather\_sunny",

],

"tree": {

"info": [

"info\_oreType\_gold",

"info\_weather\_rainy"

],

"children": [

{

"info": [

], // информация о погоде наследуется от родителя, дублировать ее не нужно

"children": [

{

"info": [

], // информация о погоде и типе руды наследуется от родителя, дублировать ее не нужно

"children": []

},

{

"info": [

"info\_oreType\_cooper",

"info\_weather\_sunny"

], // на данной планете тип руды и погода другие, не как у родителя

"children": []

}

]

}

]

}

}

Формат:

interface Result {

atomicInfo: string[]

tree: ResultNode

}

interface ResultNode {

info: string[]

children: ResultNode[]

}

**Примечание**

Файл с решением требуется оформить по шаблону:

module.exports = function (input) {

// ваше решение

return result;

}

Для построения имени характеристики нужно использовать следующий алгоритм:

func({

"color": "red",

}) === "info\_color\_red"

Не допускается использование минифицированного кода в решениях. Организаторы оставляют за собой право принимать решение о результатах соревнования на основе экспертной оценки исходного кода участников.