GRUPO 10

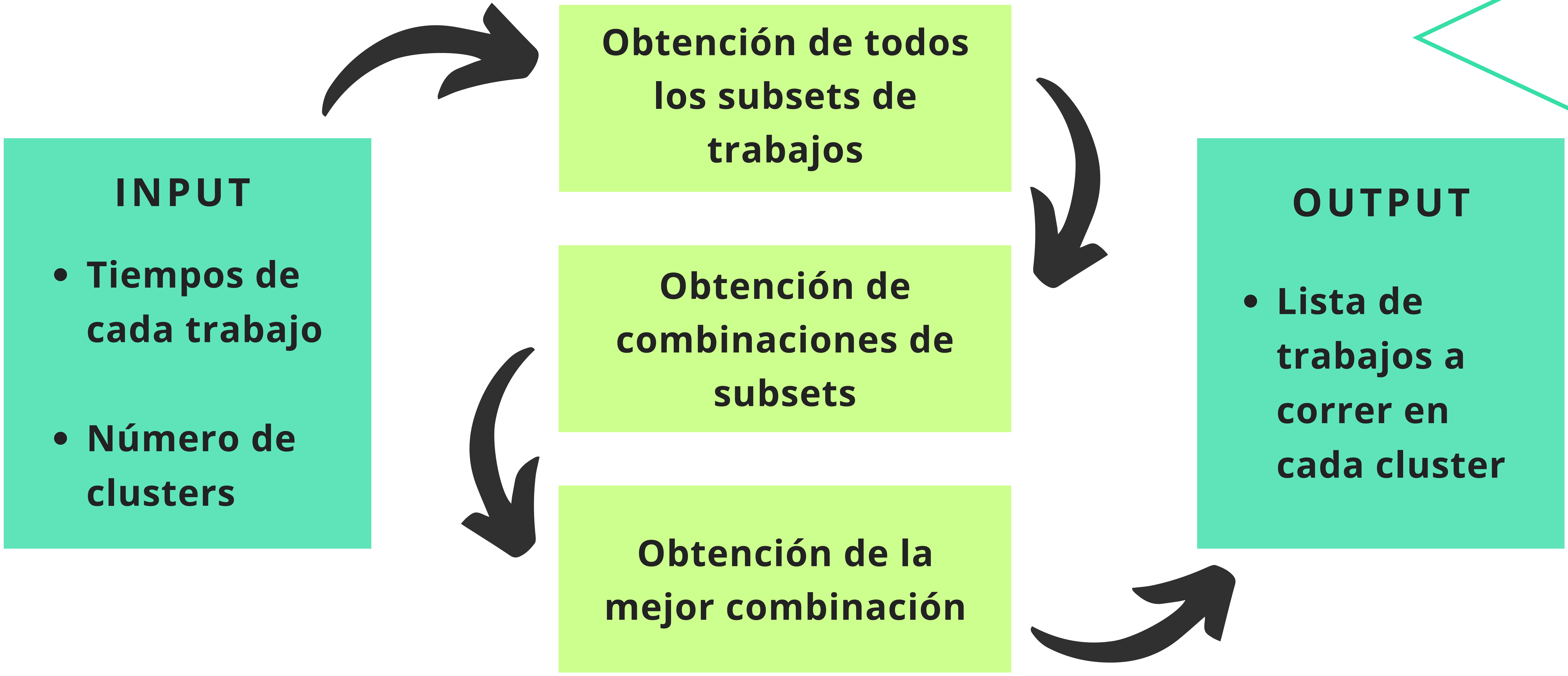
WEB ASSEMBLY

DISEÑO AVANZADO DE APLICACIONES WEB

RAFAELA KARA - CRISTOBAL MUÑOZ - FLAVIO TARSETTI

DEMO

ALGORITMO IMPLEMENTADO



INPUT

- Tiempos de cada trabajo
- Número de clusters

Obtención de todos los subsets de trabajos

Obtención de combinaciones de subsets

Obtención de la mejor combinación

OUTPUT

- Lista de trabajos a correr en cada cluster

OBTENCIÓN DE SUBSETS

INPUT

Lista con tiempos de cada trabajo

[30, 50, 10]

[]

[] [10]

[], [10] [30], [10,30]

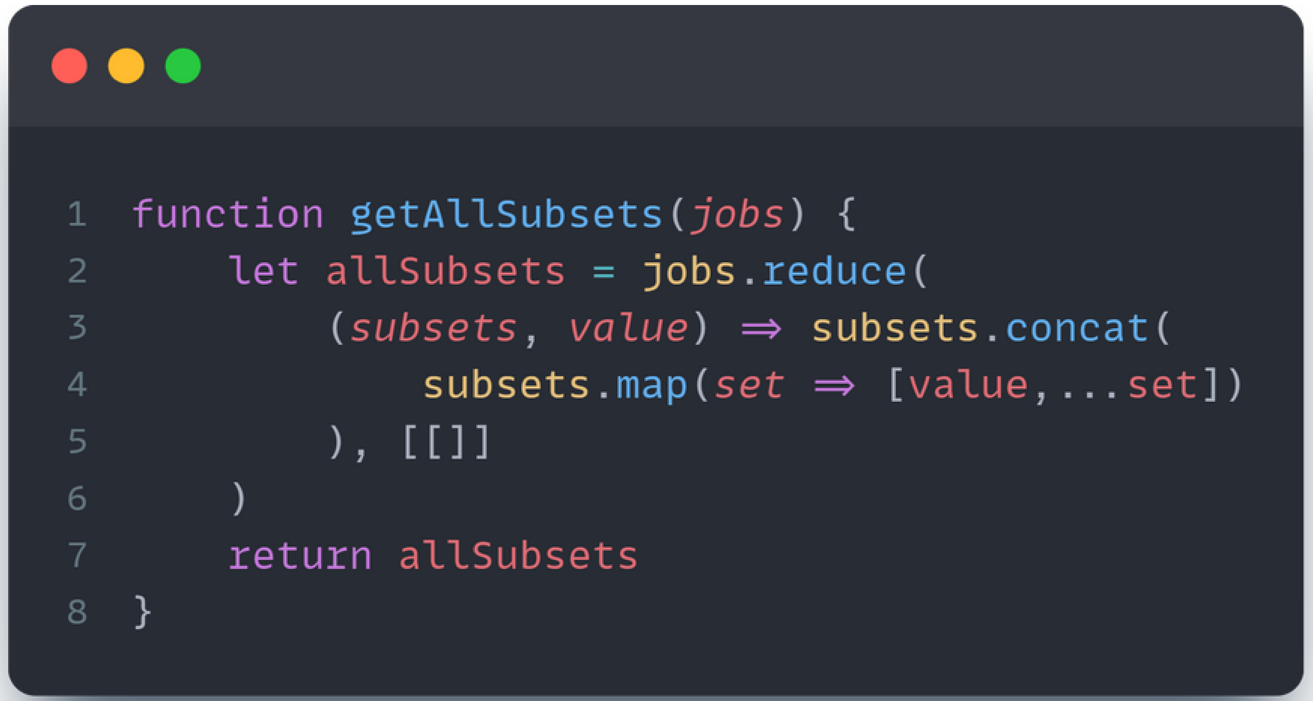
[],[10],[30], [10,30] [50],[10,50], [30,50],[10,30,50]

OUTPUT

Todos los subsets

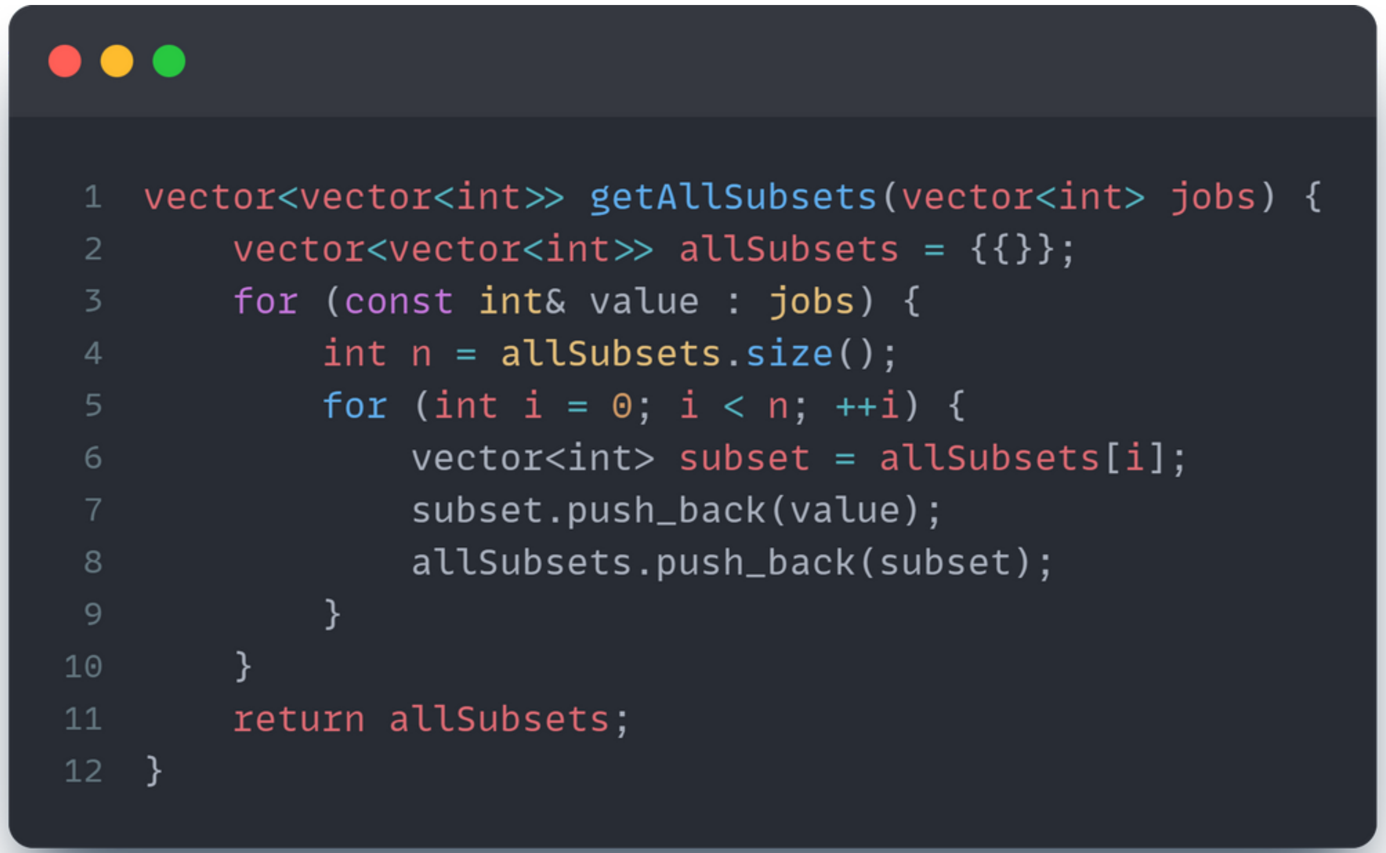
[], [10], [30], [50], [10,30], [10,50], [30,50], [10,30,50]

JS



```javascript
function getAllSubsets(jobs) {
    let allSubsets = jobs.reduce(
        (subsets, value) => subsets.concat(
            subsets.map(set => [value,...set])
        ), [[]]
    )
    return allSubsets
}
```

C++



```cpp
vector<vector<int>> getAllSubsets(vector<int> jobs) {
    vector<vector<int>> allSubsets = {{}};
    for (const int& value : jobs) {
        int n = allSubsets.size();
        for (int i = 0; i < n; ++i) {
            vector<int> subset = allSubsets[i];
            subset.push_back(value);
            allSubsets.push_back(subset);
        }
    }
    return allSubsets;
}
```

COMBINACIONES DE SUBSETS

INPUT

Subsets de trabajos

[], [10], [30], [50], [10,30], [10,50], [30,50], [10,30,50]

Número de clusters: 2

»

[],[10,30,50] ✓

[10,30],[30,50] ✗

[50],[10,30] ✓

[30],[10,50] ✓

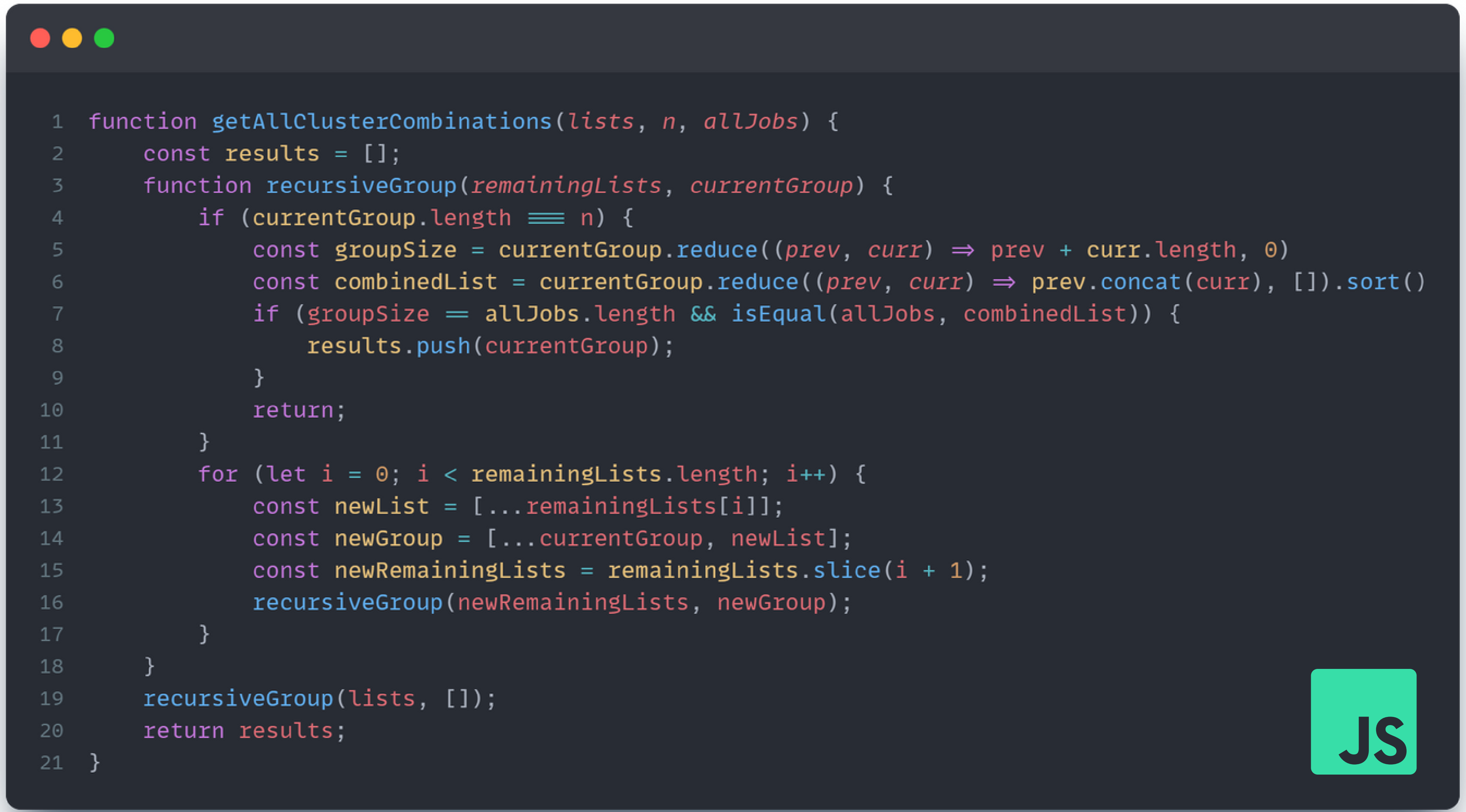
[],[10,30] ✗

»

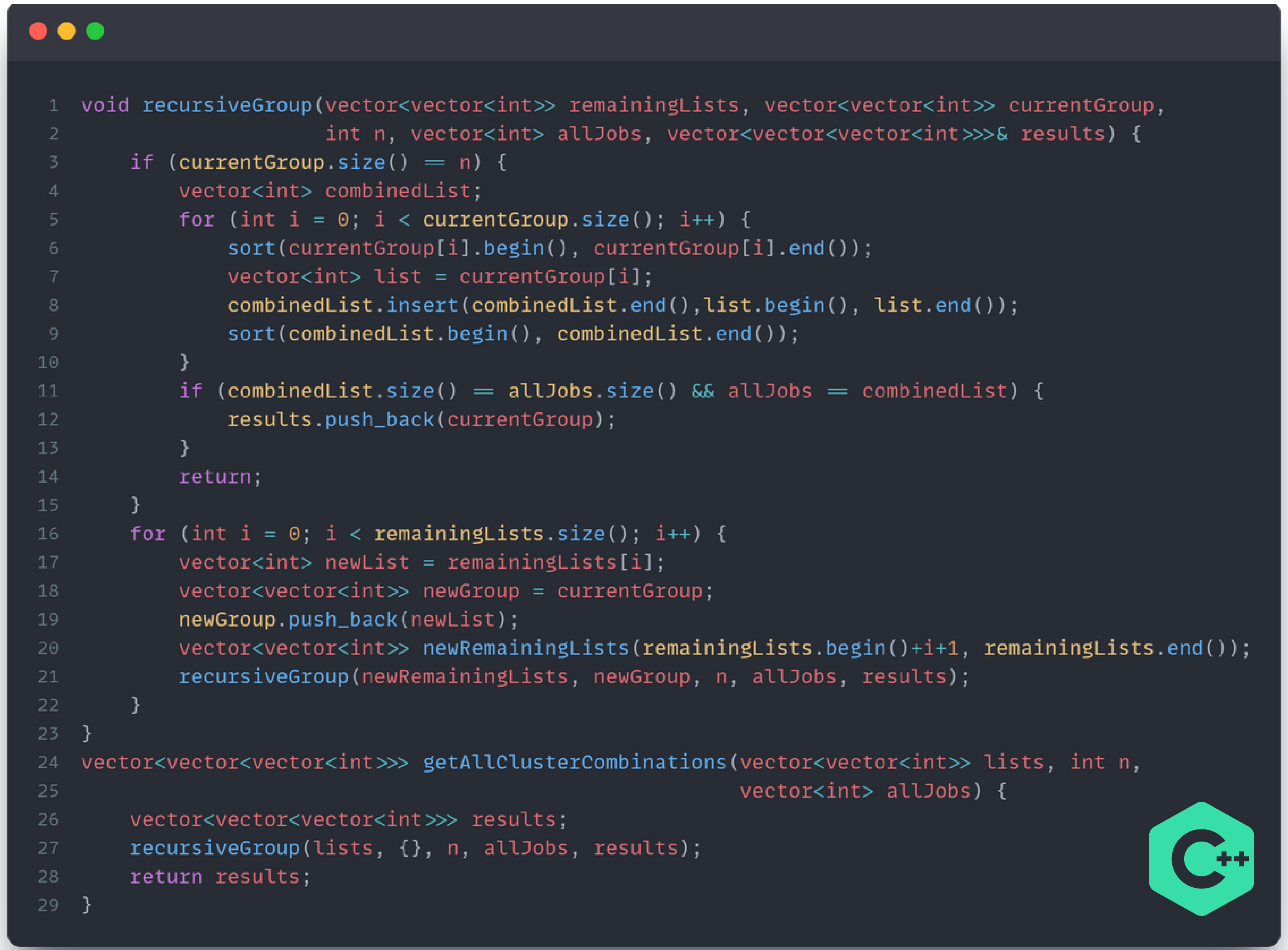
OUTPUT

Combinaciones válidas

Total: 4

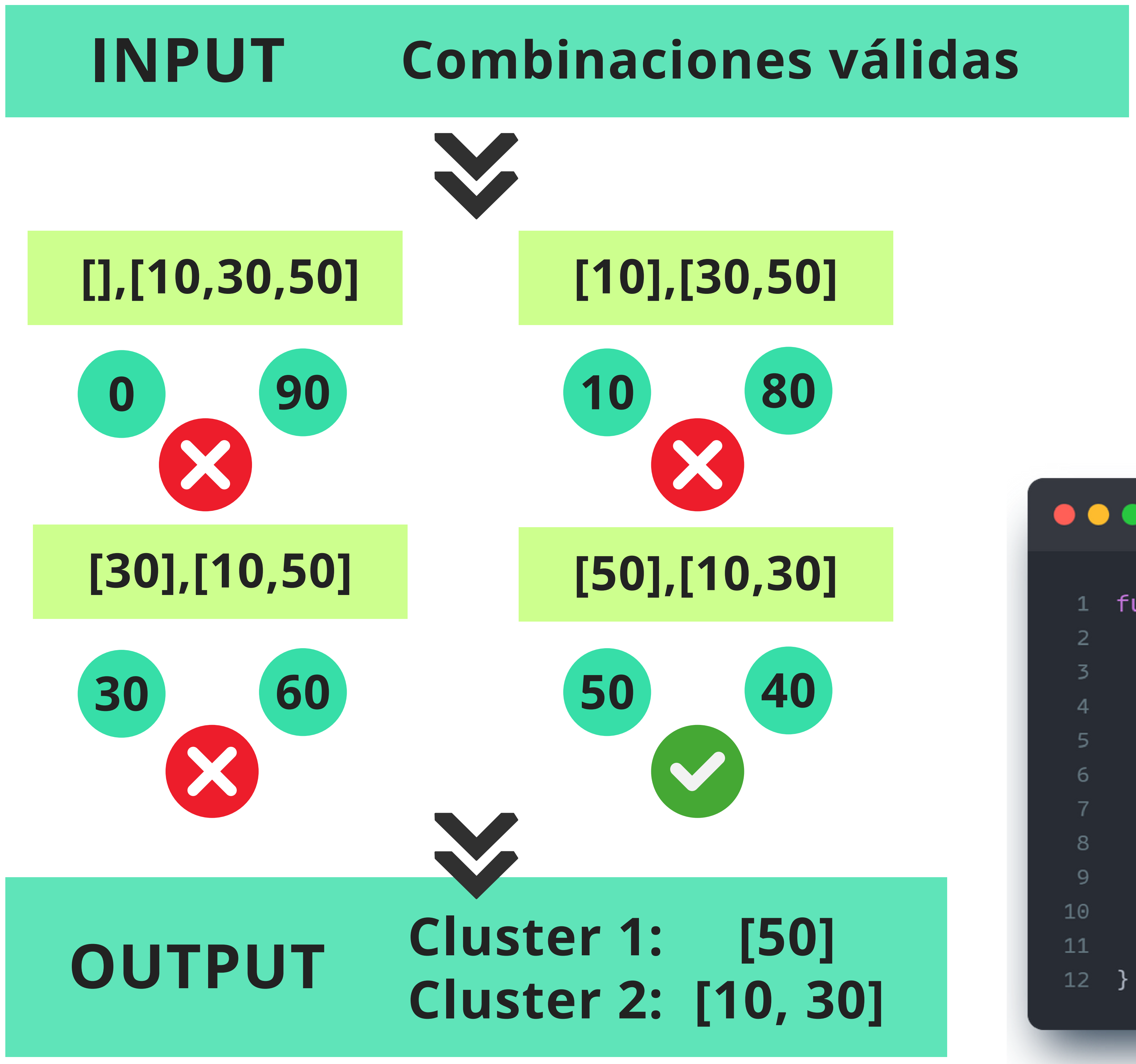


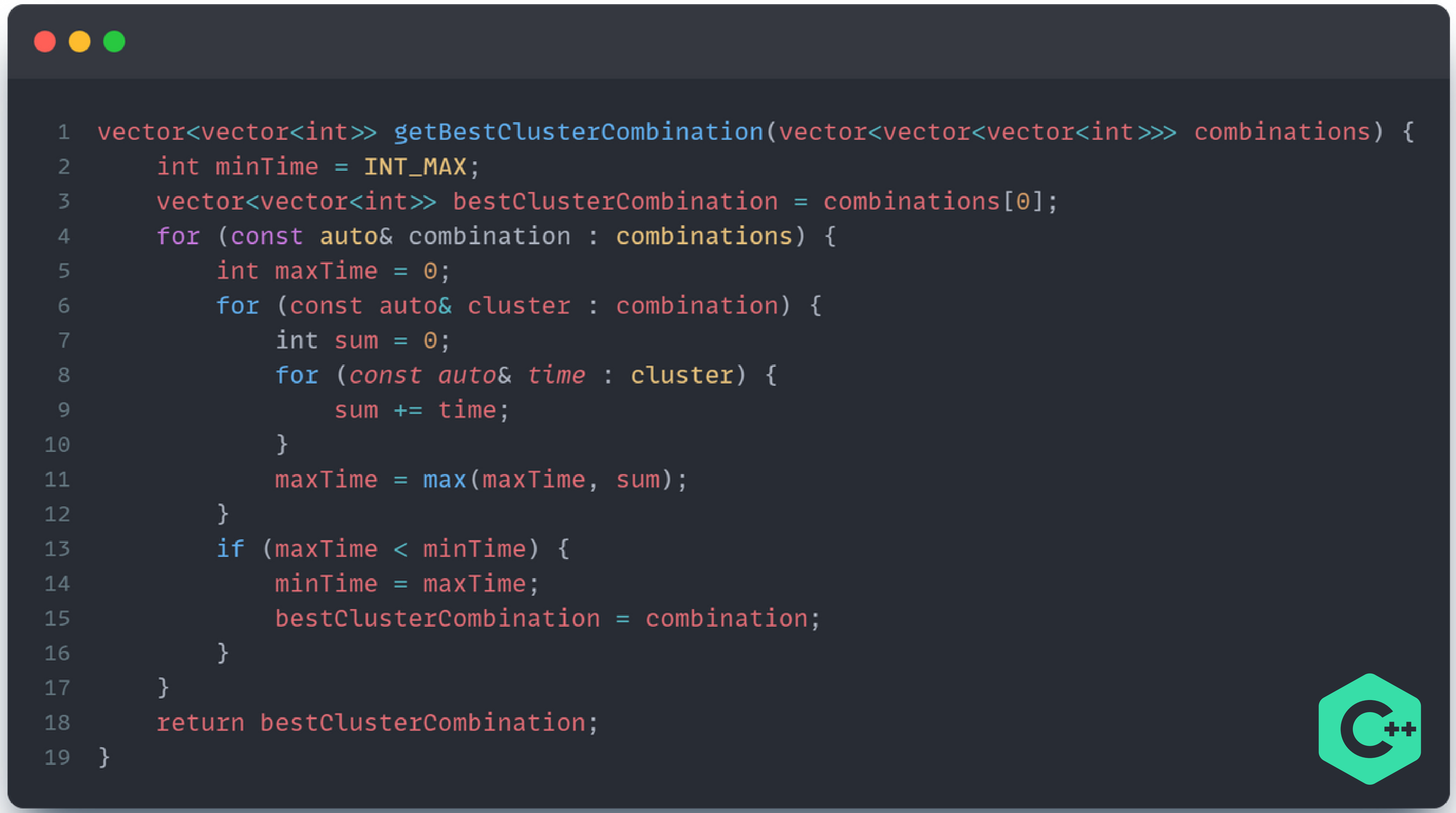
```javascript
function getAllClusterCombinations(lists, n, allJobs) {
    const results = [];
    function recursiveGroup(remainingLists, currentGroup) {
        if (currentGroup.length === n) {
            const groupSize = currentGroup.reduce((prev, curr) => prev + curr.length, 0)
            const combinedList = currentGroup.reduce((prev, curr) => prev.concat(curr), []).sort()
            if (groupSize == allJobs.length && isEqual(allJobs, combinedList)) {
                results.push(currentGroup);
            }
            return;
        }
        for (let i = 0; i < remainingLists.length; i++) {
            const newList = [...remainingLists[i]];
            const newGroup = [...currentGroup, newList];
            const newRemainingLists = remainingLists.slice(i + 1);
            recursiveGroup(newRemainingLists, newGroup);
        }
    }
    recursiveGroup(lists, []);
    return results;
}
```



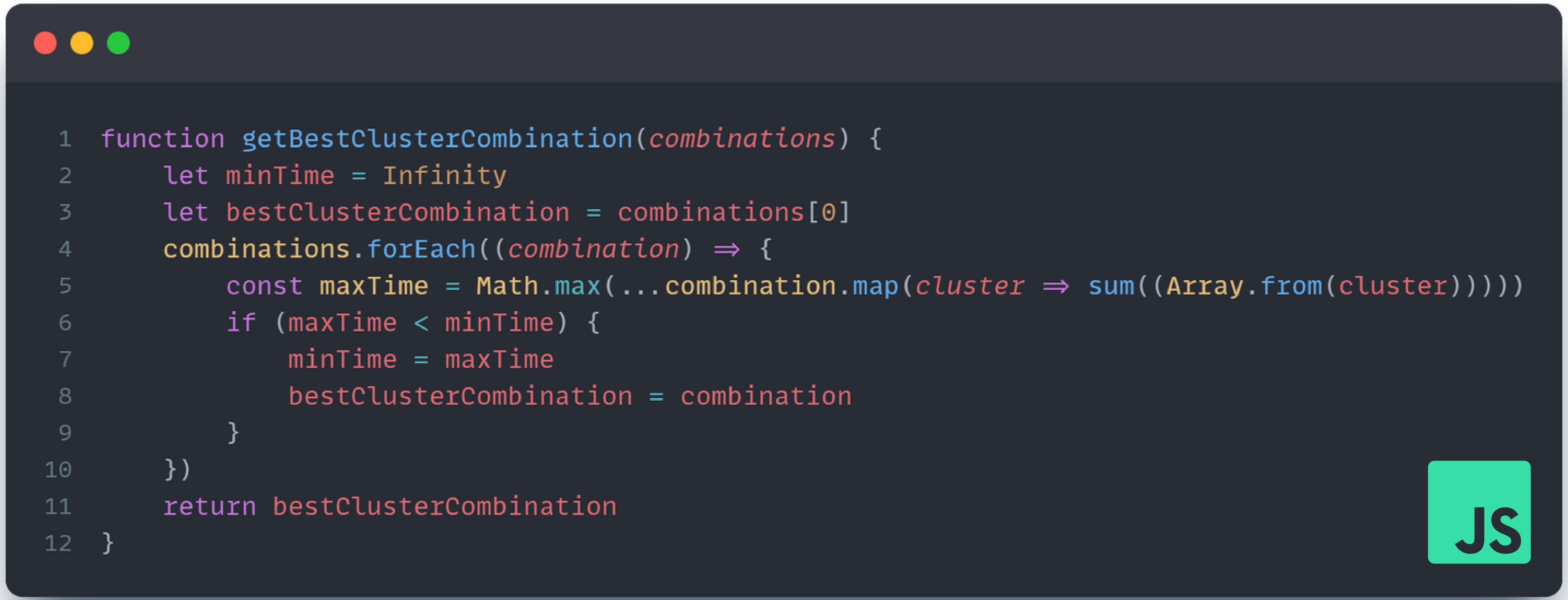
```cpp
void recursiveGroup(vector<vector<int>> remainingLists, vector<vector<int>> currentGroup,
                    int n, vector<int> allJobs, vector<vector<vector<int>>>& results) {
    if (currentGroup.size() == n) {
        vector<int> combinedList;
        for (int i = 0; i < currentGroup.size(); i++) {
            sort(currentGroup[i].begin(), currentGroup[i].end());
            vector<int> list = currentGroup[i];
            combinedList.insert(combinedList.end(),list.begin(), list.end());
            sort(combinedList.begin(), combinedList.end());
        }
        if (combinedList.size() == allJobs.size() && allJobs == combinedList) {
            results.push_back(currentGroup);
        }
        return;
    }
    for (int i = 0; i < remainingLists.size(); i++) {
        vector<int> newList = remainingLists[i];
        vector<vector<int>> newGroup = currentGroup;
        newGroup.push_back(newList);
        vector<vector<int>> newRemainingLists(remainingLists.begin()+i+1, remainingLists.end());
        recursiveGroup(newRemainingLists, newGroup, n, allJobs, results);
    }
}
vector<vector<vector<int>>> getAllClusterCombinations(vector<vector<int>> lists, int n,
                                                      vector<int> allJobs) {
    vector<vector<vector<int>>> results;
    recursiveGroup(lists, {}, n, allJobs, results);
    return results;
}
```

MEJOR COMBINACIÓN



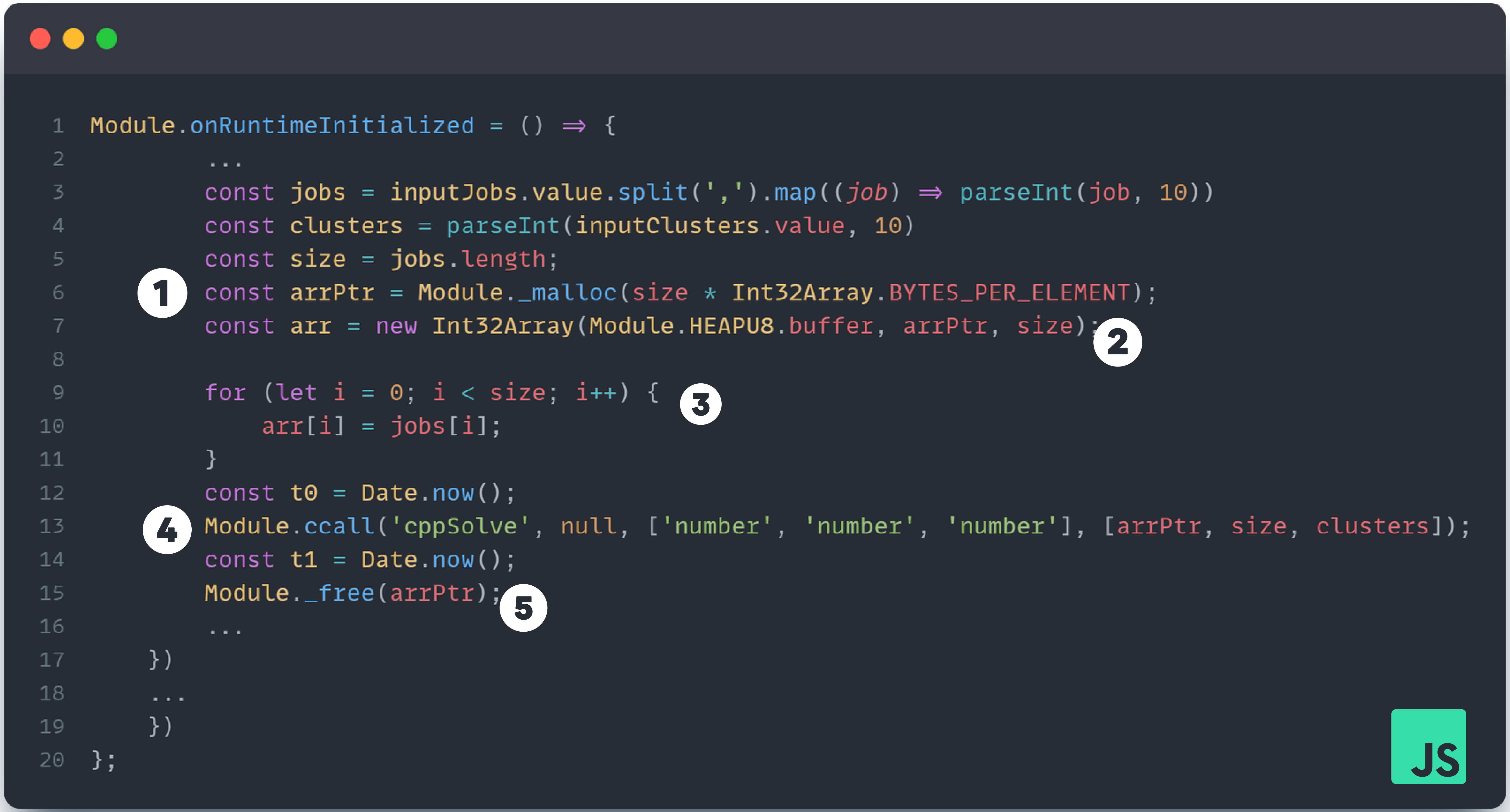


```cpp
vector<vector<int>> getBestClusterCombination(vector<vector<vector<int>>> combinations) {
    int minTime = INT_MAX;
    vector<vector<int>> bestClusterCombination = combinations[0];
    for (const auto& combination : combinations) {
        int maxTime = 0;
        for (const auto& cluster : combination) {
            int sum = 0;
            for (const auto& time : cluster) {
                sum += time;
            }
            maxTime = max(maxTime, sum);
        }
        if (maxTime < minTime) {
            minTime = maxTime;
            bestClusterCombination = combination;
        }
    }
    return bestClusterCombination;
}
```



```javascript
function getBestClusterCombination(combinations) {
    let minTime = Infinity
    let bestClusterCombination = combinations[0]
    combinations.forEach((combination) => {
        const maxTime = Math.max(...combination.map(cluster => sum((Array.from(cluster)))))
        if (maxTime < minTime) {
            minTime = maxTime
            bestClusterCombination = combination
        }
    })
    return bestClusterCombination
}
```

WASM



```
Module.onRuntimeInitialized = () => {
        ...
        const jobs = inputJobs.value.split(',').map((job) => parseInt(job, 10))
        const clusters = parseInt(inputClusters.value, 10)
        const size = jobs.length;
        const arrPtr = Module._malloc(size * Int32Array.BYTES_PER_ELEMENT);
        const arr = new Int32Array(Module.HEAPU8.buffer, arrPtr, size);

        for (let i = 0; i < size; i++) {
            arr[i] = jobs[i];
        }
        const t0 = Date.now();
        Module.ccall('cppSolve', null, ['number', 'number', 'number'], [arrPtr, size, clusters]);
        const t1 = Date.now();
        Module._free(arrPtr);
        ...
    })
    ...
    })
};
```

1. Asigna a arrPtr espacio suficiente en el Heap para almacenar todos los trabajos y retorna el puntero.
2. arr accede al array mediante la dirección del puntero en el Heap.
3. Se le asignan los valores del array jobs al array arr.
4. Se llama a la función cppSolve, definida en el programa de c++, mediante la función ccall.
5. Se libera la memoria asignada en el Heap

DIFICULTADES

- Encontrar un algoritmo que siempre sea correcto.
- Manejar memoria de subsets en C.
- Entender como funciona C++.
- Ver cómo correr el código de C++ en JavaScript usando WASM.

MUCHAS GRACIAS