class HashTable {

    constructor(size = 7) {

        this.dataMap = new Array(size);

    }

    \_hash(key) {

        let hash = 0;

        for (let i = 0; i < key.length; i++) {

            hash = (hash + key.charCodeAt(i) \* 23) % this.dataMap.length;

        }

        return hash;

    }

    printTable() {

        for (let i = 0; i < this.dataMap.length; i++) {

            console.log(i, ": ", this.dataMap[i]);

        }

    }

    set(key, value) {

        let index = this.\_hash(key);

        if (!this.dataMap[index]) this.dataMap[index] = [];

        this.dataMap[index].push([key, value]);

        return this;

    }

    get(key) {

        let index = this.\_hash(key);

        if (this.dataMap[index]) {

            for (let i = 0; i < this.dataMap[index].length; i++) {

                if (this.dataMap[index][i][0] === key) {

                    return this.dataMap[index][i][1];

                }

            }

        }

        return undefined;

    }

    keys() {

        let allKeys = [];

        for (let i = 0; i < this.dataMap.length; i++) {

            if (this.dataMap[i]) {

                for (let j = 0; j < this.dataMap[i].length; j++) {

                    allKeys.push(this.dataMap[i][j][0]);

                }

            }

        }

        return allKeys;

    }

}

Interview 1

function itemInCommon(arr1, arr2) {

    for (let i = 0; i < arr1.length; i++) {

        for (let j = 0; j < arr2.length; j++) {

            if (arr1[i] === arr2[j]) return true

        }

    }

    return false

}

**Interview 2**

function itemInCommon(arr1, arr2) {

  let obj = {}

  for (let i = 0; i < arr1.length; i++) {

    obj[arr1[i]] = true

  }

  for (let j = 0; j < arr2.length; j++) {

    if (obj[arr2[j]]) return true

  }

  return false

}

**Set Method**

///  WRITE SET METHOD HERE  ///

set(key, value) {

    // \_hash() the key to get an index value of where to store the data in memory

    let index = this.\_hash(key);

    // If there is no data in the address, create a new empty array[] to place the data in.

    if (!this.dataMap[index]) {

        this.dataMap[index] = [];

    }

    // Othervise, push() the new [key, value] pair.

    this.dataMap[index].push([key, value]);

    // Return the hash table.

    return this;

}

**Get Method**

///  WRITE GET METHOD HERE  ///

get(key) {

    // \_hash() the key to get an index

    let index = this.\_hash(key);

    // If there is an item at the given index, then map over each [key, value] pair in that index

    if (this.dataMap[index]) {

        for (let i = 0; i < this.dataMap[index].length; i++) {

            // If the key in the index matches the key we are looking for,

            // return that key

            if (this.dataMap[index][i][0] === key) {

                return this.dataMap[index][i][1];

            }

        }

    }

    // If the key is not found, return undefined

    return undefined;

}

**Keys**

///  WRITE KEYS METHOD HERE  ///

keys() {

    // Create an empty array[] that will return all of our KEYS

    let allKeys = [];

    // Map through the entire hash HashTable

    for (let i = 0; i < this.dataMap.length; i++) {

        // If an address contains data:

        if (this.data[i]) {

            // Map over each item in that memory location

            for (let j = 0; j < this.dataMap[i].length; j++) {

                // push() the keys from each [key, value] pairs into the allKeys[] array

                allKeys.push(this.dataMap[i][j][0]);

            }

        }

    }

    // Return the allkeys[] array

    return allkeys;

}