const \_ = {

clamp(number, lower, upper){

let lowerClampedValue = Math.max(number,lower);

let clampedValue = Math.min(lowerClampedValue,upper);

return clampedValue;

},

inRange(number ,start, end){

if (end == undefined){

end = start

start = 0

}

if(start > end){

var temp = end

end = start

start = temp

}

var isInRange = start <= number && number < end

return isInRange

},

words( string){

var words = string.split (' ');

return words;

},

pad(string,length){

if(length <= string.length) return string;

var startPaddingLength = Math.floor((length - string.length) / 2);

var endPaddingLength = (length - string.length - startPaddingLength);

var paddedString = ' '.repeat(startPaddingLength);

paddedString += string;

paddedString += ' '.repeat(endPaddingLength);

return paddedString;

},

has(object,key){

return object[key] != undefined;

},

invert(object) {

let keys = Object.keys(object);

let values = keys.map(k => object[k]);

let ret = {};

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

ret[values[i]] = keys[i];

}

return ret;

},

findKey(object, fn) {

for (let key in object) {

const value = object[key];

const fnRet = fn(value);

if (fnRet) return key;

}

},

drop(arr, number = 1) {

let ret = [];

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

if (i >= number) ret.push(arr[i]);

}

return ret;

},

dropWhile(arr, predicate){

let dropNumber = arr.findIndex(function(el, index, arr) {

return !(predicate(el, index, arr));

})

let droppedArray = this.drop(arr, dropNumber);

return droppedArray;

},

chunk (array, size = 1){

let arrayChunks = [];

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

let arrayChunk = array.slice(i, i+size);

arrayChunks.push(arrayChunk);

}

return arrayChunks

},

}

// Do not write or modify code below this line.

module.exports = \_;