Starter Environment: <https://typescriptlang.org/play/>

With code from class: <https://www.typescriptlang.org/play/?#-hDCMMBDDAWFaCYyCJEuqQwKeNHFPkI74DAABKoHgSYKwMSALqwPyiEATAIAXJRLYmNoOEwFEVQrKIJg3lW4D4GQREwAAAl6aDIOwIntp2pEwCU+TjjRqbaCZdFcV0BDKcRqkgVAYGQDA-GxEJOnoU02GILhUl0bJOKVneYBKURZzyUFnlaGhrZUROpF9rFNB-khgHKcOOA2mOsV6aigyIi5qAodFmFeRG4zkIi9mOc5-7IahsUlVoOHkLO6iwMAYEgX6xWkRJWDaNFK4ppuKoZQmOgdagXVQD1pVXggyAQCYTbhBGiB+tSiIFgAbss+TyGAWmhO5WgVUCO17TA8haKgyARgAmsQyUAfQVo4M24RutcVCHZwCUtgcN13Y9aDPTwg4yn5ED4Ndt0PcQ6RA-DaBnbyMnQwdR3pL9iCozODCIiEsA47+J2bTKVXgVDMM43jyqIhATIk7t8DhMd44QzK6M01jWAs2zdME0CRMwEjIOoKTBXk7ylM1tzsPA8QgtpYCjOwGLxBUPzx0FZzaOlBjGtoOk2vK1tyS7dwMD0Sm65ltLtKMX0ujJhIOKoJITyQKIa2CcJnEOM41M40HjgLUt2Z+Ii9NAhAyAVK7IDRn8kPczbJYZmWgutaqsDQKt61QBA9grWtfoANoALrpFX9C500yCIPihml4X2OIFqMCt36egYY3+K6Ac7yyDMfj1xcSDN-R3ewFgdSdzPegAGKT4PbyKCP5YsPnZdF-gXrhBAxS3P3LcObv0474X0MH0fVAT03Z8F3683vZRz9QAA4nDiBLF67edz-s8GeBxxwAD5aD0Cvn6b+d1f7IC9PgNWJ9H7T3PoXA4uhBAAFoRrQK-j-P+SCuD30nk-XemDBBOBGjSfBsC1pELWoERAKCp6IBntOJ2WUP70PgYgomrDyGF0vPQBg4VqyRUEMGGevCIDJxVu-XevCiG4gAKLIGAMfKgSNECgPQS-TAYCHZvSsqOGKH98AhxcLoewjgXA6KjgqPwOc374JMFgQQU1474AuB7dgVA6GEIQfgAA1ogCociRHYm0FI+AMif6RPoGoLhOh4CnzQRYnG040moPYfoqAljeYOEEPiIMDRRqmJRKk9JeTMlY2sYDOG4Q9q4RSIHFwOS2Ez3wEbXwSTLhkIybvJB8dE6QOxOHEwAA1RAwAMgxlNICfEYNSBDkBBUFZCIgSCk2dKdKbBUCfVPBLZZNUUoHA2WcgCBwdlXPBgsyG1NCRYHxFs-Wsl8AsmZG82k3NhRYEFD81qiJTyDSSmTICkM1CyxgAAAxmV9VAGQqAABIfjc3xBiA4aLuYVCxTAHFBt8CCgxDQWFPzKpsRrLoBFxzkV2KeaHTBQdcVMvadTYU1iaDlNjPjB0Ey36Ch4CABFACYAIuRbIA4ihpWcLGmYqgQqPYIpPGeWKtAt4wCbl6a47U37sAqEvQK1Z3EPMRDE6R+TZF7DWZFaKNreWUFQA6vMDBKhenyDPAAIssCiWAU7Cx-v9dsNr8pHWDakUNgbFagzuc6hRskADCnVC7wAqIuD+9goggG6MgMANrlT13wT6qAyA-5iqASAru+SS3IAgYMItNbfXpGACBPa3rfWCKGRg4puCDgXGQPARmfhi2+qIcgzpQi-R6I-rWwY4jPStpQNwDtpaAn5JLvkxpcCI1aAOEmlNfo00Zt3lmnNSA83gtXRRANgIYW3tpDo+wOio28hxglOpnBX2PrhuLD9wzenfsRAeqaqb00z3sCB6ax6QEKKxK1Sd3a-Q9N-QjWYG8ABsjRRDCS7bUgDqHjavGHgAVkHI6+VVS8PdKsYHfyFFs3eGdZ5eAI1+VqCDIzM1+y1RUcQ-hwuhTOChyeORUUObODMYkKx6cLi2ANybte9Io7S1EIEfxjh2HcOdOveRxZMAAj92vShuBBQABC5TbU4YFEZptpaTO+wwE53tOZuO8hSXoEMKAQld2QJ9RAkcgW2s4-kazrDjM6MwM5rB1D4gJB03Z5ADn8hOawFg1zkKZQed0AAOVUfl5iXq2P6cVLSELTxtO2dnb65LMBMzRaXo4AAZE1hT+IIs-zqw13BMAWttY66Z+raWl7UL0487LhRUBVH9GgNaqBiu0lK4icrmX3OUedqxQdaaAD8LEUDBh2zARN3hBI1jAI+fAl2FvRxpAhpkuBOBmfPSgGsWAPq41od4J7ubXt1Ye4gb7F6wCiIqbxvoYJVi3TABGSQGq66uNLeUQoohhNYBLT0c7Wpxn10OWAJe1x9CmpmEoFQArKl9GgGgKAyPOCap1OIfI66qc0+hDAXH+PUCE5gEY9HljHwarc21OAYEWfpF55j8ZFHyeJkmizzViJkBamQKIYmHcYA9kkLJFhIENDbRS0YtWLPuJQCoDrvAeuDhHCNLJhgoOggQAqK6PgXrlNcAENIFmmxvgYCMatpE40qDi-51Eh0BwSPwmxIblHiBjde8MYLp17ueCe8Wo74AzvZNh5J0aURyS35Q5h+wHLtUZAwAh-gAvsO5XS50JXovJfNU47zYXpPIRS-l4njmiWdfi8pSDoofAZGOjrcTHX1v4N+lC7r33VAIR3eLGWKsTv30e8l5UgoZQOfQeZVr83yQM+58v1EfXNJs-EC94Ap426lyCpVxrpXeg9O9QnzPxfngV-vFIFSCsGAzw5DZ71gblfxLw-wqH3kPmPlPxCDf1QFAN6HWi0BWGcTEXpEZFgHuH6jQGgJAK8TAK-0QOKBgGwUOGJHuCMUeFEGwWwUAPrgACsF9wRl8JZy869oxiIqB7gYQ8ogRy4oDz8cDr9y5RBK4Dg+CYDQDy5aDK5q4sBeCsD+CUoJCpDRD5DxDcChDpCzpC0R9qlgDFDcCx4dCX9sD9Dr98AUg1p8QUIqArdrdyAgA>

The Code:

const clear = console.clear;

clear();

// const metersInAKilometer = 1\_000;

// console.log('Meters In A Kilometer: ' + metersInAKilometer);

// let age: number | string = 30;

// age = 'Thirty One';

// let age = 30;

// console.log(age);

/\*

age++;

age += 2;

age = Math.min((age + 1), 100);

\*/

// for (let i = 0; i < 200; i++) {

// age = Math.min((age + 1), 100);

// console.log(age);

// }

// for (let x = 0; x < 10; x++) {

// for (let y = 0; y < 10; y++) {

// for (let z = 0; z < 10; z++) {

// }

// }

// }

// console.log(age);

// while (age < 100) {

// age++;

// console.log(age);

// }

// function printHelloWorld() {

// console.log('Hello World!');

// }

// printHelloWorld();

// function printAString(stringToPrint: string) {

// console.log(stringToPrint);

// }

// printAString('Hello World!');

// printAString((99).toString());

// function howManyMetersInAKilometer(): number {

// return metersInAKilometer;

// }

let numberOfStringsLogged = 0;

/\*\*

\* Logs a string to the console. Returns the count for number of strings logged by this function.

\* @param stringToLog The string to log to the console.

\* @returns number of strings logged by this function.

\*/

function logString(stringToLog: string): number {

console.log(stringToLog);

numberOfStringsLogged++;

return numberOfStringsLogged;

}

// const currentStringsLogged = logString('Hello');

// console.log(currentStringsLogged);

class Student {

private \_name: string;

private \_gradYear: number;

constructor(name: string, gradYear: number) {

this.\_gradYear = gradYear;

this.\_name = name;

}

get name(): string {

return this.\_name;

}

set name(value: string) {

this.\_name = value;

}

get gradYear(): number {

return this.\_gradYear;

}

set gradYear(value: number) {

this.\_gradYear = value;

}

private helloWorld() {

console.log('Hello from inside of the ' + this.name + ' class!');

}

sayHello() {

this.helloWorld();

}

}

// const students: Student[] = [];

const laexTheStudent = new Student('Laex', 2012);

const flexTheStudent = new Student('Flex', 2012);

// students.push(laexTheStudent);

// students.push(flexTheStudent);

const studentGradeMap = new Map<Student, string>();

studentGradeMap.set(laexTheStudent, 'A-');

studentGradeMap.set(flexTheStudent, 'A+');

// studentGradeMap.delete(laexTheStudent);

// console.log(studentGradeMap.get(laexTheStudent));

function logAllStudentGrades() {

studentGradeMap.forEach((grade, student) => {

console.log(student.name + ': ' + grade);

});

}

// const students = Array.from(studentGradeMap.keys());

// logAllStudentGrades();

// console.log(laexTheStudent.name);

// laexTheStudent.name = 'Axel';

// console.log(laexTheStudent.name + ', graduated in ' + laexTheStudent.gradYear);

// flexTheStudent.sayHello();

// class Vec3 {

// x: number;

// y: number;

// z: number;

// constructor(x: number, y: number, z: number) {

// this.x = x;

// this.y = y;

// this.z = z;

// }

// toString(): string {

// // return `Vector3(${this.x}, ${this.y}, ${this.z})`;

// return 'Vector3(' + this.x + ', ' + this.y + ', ' + this.z + ')';

// }

// }

// const zeroVec = new Vec3(1, 0, 0);

// console.log(zeroVec.toString());

// export const myFunctions = {

// logAllStudentGrades,

// logString,

// clear,

// }

type StudentData = {

grade: string,

name: string,

gradYear: number,

isCurrentlyAStudent: boolean,

}

const studentDataMap = new Map<Student, StudentData>();

const studentData = createStudentData(laexTheStudent, 'A-', false);

studentDataMap.set(laexTheStudent, studentData);

function createStudentData(student: Student, grade: string, isCurrentlyAStudent: boolean): StudentData {

return {

grade: grade,

name: student.name,

gradYear: student.gradYear,

isCurrentlyAStudent: isCurrentlyAStudent,

}

}

// laexTheStudent.gradYear = 2016;

// if (laexTheStudent.gradYear < 2015) {

// console.log(laexTheStudent.name + ' is a boomer, lol');

// }

// else {

// console.log(laexTheStudent.name + ' is a zoomer, lol');

// }

// const laexStudentData = studentDataMap.get(laexTheStudent);

// if (laexStudentData) {

// // laexStudentData.grade = 'B';

// if (laexStudentData.grade === 'A-') {

// console.log('Weak Sauce!');

// }

// else if (laexStudentData.grade === 'A+' || laexStudentData.grade === 'A') {

// console.log('NEEERD');

// }

// else if (laexStudentData.grade !== 'F' && laexStudentData.grade !== 'F-' && laexStudentData.grade !== 'F+') {

// console.log('Try Harder');

// }

// else {

// console.log('Really? Reallly? Come on now...');

// }

// }

// let someBoolean = true;

// someBoolean = !someBoolean;

// console.log(Math.random());

const startTime = Date.now();

const runForMs = 2\_000;

// console.log(startTime);

// while (startTime + runForMs > Date.now()) {

// const curTime = Date.now();

// console.log(curTime);

// await new Promise(resolve => setTimeout(resolve, 100));

// }

// const asyncID = setInterval(() => {

// console.log(Date.now());

// }, 500);

// setTimeout(() => {

// clearInterval(asyncID);

// }, 5\_000);

// const randomNumber = Math.random();

// console.log(randomNumber);

// const randomInteger = Math.floor(randomNumber + 0.5);

// console.log(randomInteger);

// const randomLargeInt = Math.floor(randomNumber \* 1\_000);

// console.log(randomLargeInt);

const largeNumberArray: number[] = [];

while (largeNumberArray.length <= 1\_000) {

largeNumberArray.push(largeNumberArray.length);

}

for (let i = largeNumberArray.length - 1; i > 0; i--) {

const j = Math.floor(Math.random() \* (i + 1));

[largeNumberArray[i], largeNumberArray[j]] = [largeNumberArray[j], largeNumberArray[i]];

}

console.log(largeNumberArray);

console.log(largeNumberArray.indexOf(1000));