 МІНІСТЕРСТВО ОСВІТИ I НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ   
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ БІОМЕДИЧНОЇ ІНЖЕНЕРІЇ

КАФЕДРА БІОМЕДИЧНОЇ КІБЕРНЕТИКИ

**Комп’ютерний практикум №5**

з дисципліни «Веб-технології та веб-дизайн-1»

**Виконав:**

студент гр. БС-81

Тарабара С.В.

**Перевірив:**

ас. каф. БМК Матвійчук О. В.

Зараховано від \_\_\_.\_\_\_.\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис викладача)

Київ-2020

**Завдання:**

**Завдання:**

В результаті у кожного мають бути функції для виконання 3х завдань,

1 - обов’язкове для всіх,

2 - variant mod 6

3 - (((group mod 10) + variant) mod 6) + 6

**Завдання обов'язкове для всіх:**

**Фіскальний кодекс**

Кожна особа в Італії має унікальний ідентифікаційний ідентифікаційний код, виданий національною податковою службою після реєстрації народження: Фіскальний кодекс (Codice Fiscale). [<https://en.wikipedia.org/wiki/Italian_fiscal_code_card>]

Враховуючи об'єкт, що містить особисті дані людини (ім'я, прізвище, стать та дата народження), поверніть 11 символів коду як рядок, виконуючи наступні кроки:

Створіть 3 символи прізвища, якщо воно має:

Принаймні 3 приголосні, тоді використовуються перші три приголосні. (Ньюмен -> NWM).

Менше ніж 3 приголосні, тоді голосні замінять відсутні літери в тому ж порядку, в якому вони відображаються (Fox -> FXO | Hope -> HPO).

Менше трьох літер тоді "X" займе третю позицію після приголосного та голосного (Yu -> YUX).

Створіть від імені 3 великі літери, якщо у них є:

Рівно 3 приголосних тоді приголосні вживаються в тому порядку, в якому вони з’являються (Метт -> МТТ).

Вживається більше 3 приголосних, тоді перший, третій та четвертий приголосні (Саманта -> СНТ | Томас -> ТМС).

Менше трьох приголосних, тоді голосні замінять відсутні літери в тому ж порядку, в якому вони відображаються (Bob -> BBO | Paula -> PLA).

Менше трьох літер тоді "X" займе третійю позицію після приголосного та голосного (Al -> LAX).

Утворіть 2 числа, 1 букву та 2 числа від дати народження та статі:

Візьміть дві останні цифри року народження (1985 -> 85).

Створіть лист, відповідний місяцю народження (січень -> A | грудень -> T), використовуючи таблицю для перерахунку, включену в код.

Для чоловіків приймають день народження, додаючи один нуль на початку, якщо він менше 10 (будь-який 9-й день -> 09 | будь-який 20-й день -> 20).

Для жінок беруть день народження і сумують до нього 40 (будь-який 9-й день -> 49 | будь-який 20-й день -> 60).

PS:

* Букви в відповіді мають бути в uppercase
* Дата народження в форматі D/M/YYYY
* Таблиця перетворення наведена в завданні
* Для людини створити клас
* Відображення місяця в літеру: // const months = { 1: "A", 2: "B", 3: "C", 4: "D", 5: "E", 6: "H", 7: "L", 8: "M", 9: "P", 10: "R", 11: "S", 12: "T" }

*Приклад:*

fiscalCode({

name: "Matt",

surname: "Edabit",

gender: "M",

dob: "1/1/1900"

}) ➞ "DBTMTT00A01"

fiscalCode({

name: "Helen",

surname: "Yu",

gender: "F",

dob: "1/12/1950"

}) ➞ "YUXHLN50T41"

**Завдання по варіантах:**

1. **Візерунчастий ремінець**

Браслет може мати 4 візерунка:

* горизонтальний: кожен елемент підряд однаковий.
* вертикальний: кожен елемент у стовпці однаковий.
* діагональна зліва: кожен елемент ідентичний тому, що знаходиться вгорі ліворуч або праворуч знизу.
* діагональна справа: кожен елемент ідентичний тому, що знаходиться вгорі праворуч або внизу зліва.

Напишіть функцію, яка повертає True, якщо розділ можна правильно класифікувати на один із 4 типів, в іншому випадку False.

*Приклад:*

isWristband([

["A", "A"],

["B", "B"],

["C", "C"]

]) ➞ true

// Part of horizontal wristband.

isWristband([

["A", "B"],

["A", "B"],

["A", "B"]

]) ➞ true

// Part of vertical wristband.

isWristband([

["A", "B", "C"],

["C", "A", "B"],

["B", "C", "A"],

["A", "B", "C"]

]) ➞ true

// Part of diagonal left wristband.

isWristband([

["A", "B", "C"],

["B", "C", "A"],

["C", "A", "B"],

["A", "B", "A"]

]) ➞ true

// Part of diagonal right wristband.

1. **Симетричні візерунки**

Кетлін володіє прекрасним магазином килимів. Їй подобається групувати килими в 4 взаємовиключні категорії.

* недосконалий => imperfect
* горизонтально симетричний => horizontally symmetric
* вертикально симетричний => vertically symetric
* досконалий => perfect

Недосконалий килим - це ні горизонтально, ні вертикально симетрично. Ось приклад недосконалого килима:

[

["a", "a", "a", "a"],

["a", "a", "a", "a"],

["a", "a", "b", "b"]

]

Далі - горизонтально симетричний килим. Ви можете "скласти" килим через гіпотетичну вісь x, і обидві сторони були б однаковими. Горизонтально симетричний килим не є вертикально симетричним (інакше цей килимок був би класифікований як ідеальний).

[

["c", "a", "a", "a"],

["b", "b", "b", "b"],

["c", "a", "a", "a"]

]

Далі - вертикально симетричний килим. Ви можете "скласти" килим через гіпотетичну вісь y, і обидві сторони були б однаковими. Вертикально симетричний не є горизонтально симетричним (інакше цей килимок був би класифікований як ідеальний).

[

["a", "b", "a"],

["b", "b", "b"],

["a", "b", "a"],

["a", "b", "a"]

]

Нарешті, ідеальний килим - це вертикально і горизонтально симетричний. Тобто, складені або по довжині, або по ширині дадуть дві однакові частини.

[

["a", "b", "b", "a"],

["b", "b", "b", "b"],

["a", "b", "b", "a"]

]

Давши килим розміром m x n, визначте, чи він недосконалий, горизонтально симетричний, вертикально симетричний чи ідеальний. Килимки представлені за допомогою двовимірного масиву.

*Приклад:*

classifyRug([

["a", "a"],

["a", "a"]

]) ➞ "perfect"

classifyRug([

["a", "a", "b"],

["a", "a", "a"],

["b", "a", "a"]

]) ➞ "imperfect"

classifyRug([

["b", "a"],

["b", "a"]

]) ➞ "horizontally symmetric"

classifyRug([

["a", "a"],

["b", "b"]

]) ➞ "vertically symmetric"

**Код**

**Style.css**

\* {

margin: 0;

padding: 0;

box-sizing: border-box;

}

.column {

display: flex;

justify-content: center;

align-items: center;

}

/\* Clear floats after the columns \*/

.row:after {

content: "";

display: table;

clear: both;

}

.row {

transition: 1s;

opacity: .5;

}

.row:hover {

opacity: 1;

outline: 10px solid rgba(81, 255, 0, 0.5);

outline-offset: -10px;

box-shadow: 0 0 0 0 gray inset;

}

h4 {

display: inline;

}

.columnCode {

display: flex;

justify-content: center;

height: 20%;

}

.block {

width: 460px;

/\* background-color: rgb(252, 165, 72); \*/

border-color: rgb(4, 104, 12);

border-style: solid;

padding: 20px;

/\* margin-top: 20px; \*/

clear: both;

text-align: center;

font-size: large;

word-wrap: break-word;

border-radius: 3px;

-webkit-border-radius: 3px;

-moz-border-radius: 3px;

padding: 40px;

}

form {

margin-right: 30px;

}

.form-style-2 {

margin: 25px;

/\* max-width: 500px; \*/

padding: 20px 12px 10px 20px;

font: 13px Arial, Helvetica, sans-serif;

display: flex;

flex-direction: row;

width: auto;

}

.form-style-2-heading {

font-weight: bold;

border-bottom: 2px solid #ddd;

padding-bottom: 8px;

margin-bottom: 20px;

font-size: 15px;

}

.form-style-2 label {

display: block;

margin: 0px 0px 15px 0px;

}

.form-style-2 label>span {

width: 100px;

font-weight: bold;

float: left;

padding-top: 8px;

padding-right: 5px;

}

/\* .form-style-2 span.required{

color:red;

} \*/

.form-style-2 .birthday-field {

width: 100px;

text-align: left;

}

.form-style-2 .select-field {

width: 100px;

text-align: center;

}

.form-style-2 input.input-field-matrix {

width: 420px;

word-wrap: normal;

}

.form-style-2 input.input-field {

width: 48%;

}

.form-style-2 input.input-field,

.form-style-2 input.input-field-matrix,

.form-style-2 .birthday-field,

.form-style-2 .select-field {

box-sizing: border-box;

-webkit-box-sizing: border-box;

-moz-box-sizing: border-box;

border: 1px solid #C2C2C2;

box-shadow: 1px 1px 4px #EBEBEB;

-moz-box-shadow: 1px 1px 4px #EBEBEB;

-webkit-box-shadow: 1px 1px 4px #EBEBEB;

border-radius: 3px;

-webkit-border-radius: 3px;

-moz-border-radius: 3px;

padding: 7px;

outline: none;

}

.form-style-2 .input-field:focus,

.form-style-2 .tel-number-field:focus,

.form-style-2 .select-field:focus {

border: 1px solid #0C0;

}

.form-style-2 input[type=submit],

.form-style-2 input[type=button] {

border: none;

width: 240px;

padding: 15px 20px 15px 20px;

background: rgb(255, 230, 0);

color: #fff;

font-size: larger;

font-weight: bold;

box-shadow: 1px 1px 4px #DADADA;

-moz-box-shadow: 1px 1px 4px #DADADA;

-webkit-box-shadow: 1px 1px 4px #DADADA;

border-radius: 3px;

-webkit-border-radius: 3px;

-moz-border-radius: 3px;

}

.form-style-2 input[type=submit]:hover,

.form-style-2 input[type=button]:hover {

background: rgb(4, 104, 12);

color: #fff;

}

**JS Fiscal Code**

class User {

constructor(newName, newSurname, newGender, newBirthdate) {

this.name = newName;

this.surname = newSurname;

this.gender = newGender;

this.birthdate = newBirthdate;

}

set name(newName) { // TODO: move validation up

if (this.\_validateInput(newName)) {

this.\_name = newName;

}

}

set surname(newSurname) {

if (this.\_validateInput(newSurname)) {

this.\_surname = newSurname;

}

}

set birthdate(newBirthdate) {

if (this.\_validate\_date(newBirthdate)) {

this.\_birthdate = newBirthdate;

} else {

alert("Введена некорректная дата!");

}

}

\_validateInput(value) {

if (value === "") {

alert('Input at list 1 symbol in name or surname field');

} else {

return true;

}

}

/\* Функция разбивает дату на составляющие (метод split()),

а затем выполняет проверку составляющих при помощи объекта Date и методов getFullYear(), getMonth() и getDate(). \*/

// arrD[1] -= 1 Потому что у объекта Date отсчет месяцев начинается с 0.

\_validate\_date(value) {

let arrD = value.split("/");

arrD[1] -= 1;

let d = new Date(arrD[2], arrD[1], arrD[0]);

if ((d.getFullYear() == arrD[2]) && (d.getMonth() == arrD[1]) && (d.getDate() == arrD[0])) {

return true;

} else {

return false;

}

}

surnameCode() {

let letters = this.\_separateLetters(this.\_surname);

let vowels = letters.get("vowels");

let consonants = letters.get("consonants");

return this.\_codeFromSurname(vowels, consonants);

}

nameCode() {

let letters = this.\_separateLetters(this.\_name);

let vowels = letters.get("vowels"),

consonants = letters.get("consonants");

return this.\_codeFromName(vowels, consonants);

}

\_codeFromSurname(vowels, consonants) {

let code = '';

for (let i = 0, j = 0; code.length < 3;) {

if (typeof consonants[i] !== 'undefined') {

code += consonants[i++];

} else if (typeof vowels[j] !== 'undefined') {

code += vowels[j++];

} else {

code += "X";

}

}

return code;

}

\_codeFromName(vowels, consonants) {

let code = '';

if (consonants.length == 3) {

code = consonants.slice(0, 2).join('');

} else if (consonants.length > 3) {

code += consonants[0];

code += consonants[2];

code += consonants[3];

} else if (consonants.length < 3) {

for (let i = 0, j = 0; code.length < 3;) {

if (typeof consonants[i] !== 'undefined') {

code += consonants[i++];

} else if (typeof vowels[j] !== 'undefined') {

code += vowels[j++];

} else {

code += "X";

}

}

}

return code;

}

// Наличие символа в массиве гласных можно проверить с помощью indexOf

\_isVowel(char) {

return ['a', 'e', 'i', 'o', 'u'].indexOf(char.toLowerCase()) !== -1; // todo: arr.includes(a);

}

\_separateLetters(someData) {

let vowels = [];

let consonants = [];

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

// метод charAt() возвращает символ по заданному индексу внутри строки

let char = someData.charAt(i);

if (this.\_isVowel(char)) {

vowels.push(char);

} else {

consonants.push(char);

}

}

// Используется Map(), чтобы вернуть два массива.

let letters = new Map();

letters.set("vowels", vowels);

letters.set("consonants", consonants);

return letters;

}

/\* Функция parseInt(строка, основание) выполняет синтаксический разбор строки начиная с первого символа,

если первый символ является цифрой или знаком (- или +),

то она переходит к обработке второго символа и так далее,

пока не будет обнаружен символ, который не может быть преобразован в числовое значение,

после этого она возвращает полученное целое число \*/

\_parseDate(date) { // TODO: no variables from upper case

const parsedDate = date.split("/").map(function (value) {

return parseInt(value, 10);

})

return parsedDate;

}

\_codeFromBirthDate() {

const months = {

1: "A",

2: "B",

3: "C",

4: "D",

5: "E",

6: "F",

7: "G",

8: "H",

9: "I",

10: "G",

11: "K",

12: "T"

};

let date = this.\_parseDate(this.\_birthdate);

// adding '0' to yearCode/dayCode if there is only 1 number

let yearCode = date[2] % 100;

yearCode = yearCode < 10 ? '0' + yearCode : yearCode;

let dayCode = this.gender === "Female" ? date[0] + 40 : date[0];

dayCode = dayCode < 10 ? '0' + dayCode : dayCode;

return yearCode + months[date[1]] + dayCode;

}

getCode() {

return (this.surnameCode().toUpperCase() + this.nameCode().toUpperCase() +

this.\_codeFromBirthDate()).toUpperCase();

}

}

function makeCode() {

let user = new User(document.getElementById('name').value,

document.getElementById('surname').value,

document.getElementById('gender').value,

document.getElementById('bday').value);

document.getElementById('fiscal-code').innerHTML = user.getCode();

}

**5 Task**

function isWristband(arr) {

// check horizontal

let horizontal = true

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

for (j = 0; j < arr[0].length; j++) {

if (arr[i][j] !== arr[i][0]) horizontal = false

}

}

if (horizontal) return true

// check vertical

let vertical = true

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

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

if (arr[j][i] !== arr[0][i]) vertical = false

}

}

if (vertical) return true

// check upper left / lower right diagonal

let leftdiag = true

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

for (j = 0; j < arr[i].length; j++) {

if (arr[i][j] !== arr[(i + 1) % arr.length][(j + 1) % arr[0].length]) leftdiag = false

}

}

if (leftdiag) return true

let rightdiag = true

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

for (j = 0; j < arr[i].length; j++) {

if (arr[i][j] !== arr[(i + 1) % arr.length][j - 1 < 0 ? arr.length - 1 : (j - 1) % arr[0].length]) rightdiag = false

}

}

if (rightdiag) return true

return false

}

console.log(isWristband([

["A", "A"],

["B", "B"],

["C", "C"]

]))

// Part of horizontal wristband.

isWristband([

["A", "B"],

["A", "B"],

["A", "B"]

])

// Part of vertical wristband.

isWristband([

["A", "B", "C"],

["C", "A", "B"],

["B", "C", "A"],

["A", "B", "C"]

])

// Part of diagonal left wristband.

isWristband([

["A", "B", "C"],

["B", "C", "A"],

["C", "A", "B"],

["A", "B", "A"]

])

// Part of diagonal right wristband.

**9 Task**

function classifyRug(str) {

var Rug = str;

function symmetryHorizontal() {

for (let i = 0; i < Math.floor(Rug.length / 2); i++) {

if (Rug.length % 2 != 0) {

if (i == Math.floor(Rug.length / 2))

continue;

}

for (let j = 0; j < Rug[0].length; j++) {

if (Rug[i][j] !== Rug[Rug.length - 1 - i][j]) {

return false;

}

}

}

return true;

}

function symmetryVertical() {

for (let j = 0; j < Math.floor(Rug[0].length / 2); j++) {

if (Rug[0].length % 2 != 0) {

if (j == Math.floor(Rug[0].length / 2))

continue;

}

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

if (Rug[i][j] !== Rug[i][Rug[0].length - 1 - j])

return false;

}

}

return true;

}

if (symmetryHorizontal() && symmetryVertical())

return "perfect rug";

else if (symmetryHorizontal())

return "horizontally symmetric";

else if (symmetryVertical())

return "vertically symmetric";

else return "imperfect";

}

console.log(classifyRug([

["a", "a"],

["a", "a"]

]))

console.log(classifyRug([

["a", "a", "b"],

["a", "a", "a"],

["b", "a", "a"]

]))

console.log(

classifyRug([

["b", "a"],

["b", "a"]

]))

console.log(classifyRug([

["a", "a"],

["b", "b"]

]))