Examen Programming Fundamentals

Bachelor in de Toegepaste Informatica	
Academiejaar	2022-2023
Semester	1
Zittijd	1
Module	Programming Fundamentals
Datum en Tijd	10/01/2022 08:30 → 12:30
Lector	De Wael Mattias Vandycke Dirk (TI-Engels)
Nagelezen door	Mourisse Dieter

Bronbestanden

Alle benodigde bronbestanden worden gevonden samen bij dit bestand in het ZIP-archief. De **map** plakken in je oefeningen-repo, laat je toe op een eenvoudige manier de opgave te lezen, je examen te maken, en de testen uit te voeren. Deze aanpak heb je normaal al eens geoefend.

Je mag enkel de drie "v-files" aanpassen. Enkel bestanden met de juiste naam worden geëvalueerd.

Doel is de testen te laten runnen maar ook de stijlregels voor code te respecteren. Bij de testen kun je extra informatie vinden over doel, functienamen, parameters en uitzonderingsgevallen.

Indienen

Comprimeer de vier oplossingsfiles `v1.js`, `v2.js`, `v3.js`, `v4.js` tot één ZIP en laad ze op op LEHO. Voeg ook al deze files en de testen toe aan je git-repo, commit en push. Teken bij het verlaten van het lokaal af op de aanwezigheidslijst.

Score

Samen met het stuk op papier, maakt dit examen 40% uit van de evaluatie voor de module **Programming Fundamentals**. We scoren deze 40% op 20 punten. Aan dit deel zijn 14 punten toegekend. De overige 6 punten staan op het deel op papier.

In elke file staat de opgave die in die file gemaakt moet worden uitgeschreven. In dit bestand vind je diezelfde tekst ook terug. Onderaan elke file zie je de puntenverdeling in meer detail. Bij elke vraag wordt gescoord op het gebruik van de juiste structuur, techniek en stijl. Het is dus niet omdat de unit-test slaagt, dat je alle punten krijgt. Ook omgekeerd: panikeer niet als je de unit-test niet 100% kan doen slagen.

Toegelaten hulpmiddelen:

Laptop, balpen, potlood en gom. NIETS anders. Je mag gebruik maken van je eigen klasnotities, slides, boeken, syllabi en andere materialen op je eigen toestel. **Gebruik van het internet of elk ander soort netwerk is strikt verboden.** Enkel voor het ophalen en indienen van de oplossing mag verbinding gemaakt worden met Leho.

Communicatie met medestudenten of derde partijen (AI en/of RI) is strikt verboden (zie hierboven). Zorg ervoor dat alle applicaties die mogelijks zouden openspringen/opstarten of geïnterpreteerd zouden kunnen worden als een poging tot fraude (CoPilot, Outlook, Messenger, Facebook, …) afgesloten zijn! Sluit alle applicaties volledig af die de schijn van (online) communicatie kunnen wekken (niet limitatief: Facebook (messenger), mail client, ...). Enkel WebStrom (of één andere IDE) dient open te staan.

Alles wat lokaal op je eigen laptop is opgeslagen en/of geïnstalleerd en dat geen verbinding nodig heeft met enigerlei netwerk (niet limitatief: geen wifi, geen ethernet, geen bluetooth, geen IR, ...) mag gebruikt worden. Audio is niet toegelaten.

Iedere vaststelling van onregelmatigheid (o.a. GSM, spieken, afkijken) wordt conform het OER gemeld aan de betrokken student en aan de voorzitter van de examencommissie.

Let op: Enkel bestudeerde syntax en technieken gebruiken

Veel van onderstaande opgaven kan je in JavaScript gemakkelijk als "one-liner" schrijven door gebruik te maken van ingebouwde functies of *syntactic sugar*. Maak in deze opdracht enkel gebruik van bestudeerde technieken. *Als je twijfelt, schijf je de gewenste functionaliteit uit als hulpfunctie.*

Mag dus niet gebruikt worden (niet-limitatieve opsomming): `slice`, *array spreading*, *object destructuring*, `indexOf`, `find`, `map`, `filter`, `reduce`, ...

Houd als vuistregel max. 7 lijnen code per functie aan. Schrijf dus hulpfuncties waar nodig en verzorg de naamgeving, want ook hier wordt op gequoteerd.

V1: Split

Schrijf de functies `take` en `drop` die beiden een array en een (geheel) getal verwachten als input. `take` geeft de eerste `n` elementen uit die array terug, `drop` geeft alles *behalve* die eerste `n` elementen uit die array terug.

Bekijk de testen om na te gaan wat je moet doen in de randgevallen.

Schrijf nu ook de functie `split` die een array opsplitst in gelijke deeltjes. Doe dit door zo weinig mogelijk nieuwe code te schrijven.

```
> split(["a", "b", "c", "d", "e", "f"], 2);
[["a", "b"], ["c", "d"], ["e", "f"]]

> split(["a", "b", "c", "d", "e"], 2);
[["a", "b"], ["c", "d"], ["e"]]);
```

Als het niet lukt om `split` werkende te krijgen en je zou split later graag nog hergebruiken, mag je volgende code gebruiken als body voor `split`, je krijgt dan natuurlijk geen punten voor dit deel: `return new Array(Math.ceil(arr.length / n)).fill(0).map(()=>arr.splice(0, n));`

V2: Objects and Arrays:

- Schrijf een functie die van een array van gelijkaardige objecten een nieuwe array maakt, met daarin enkel de waardes van één bepaalde *property.* (in de testen zie je een voorbeeld met een array van kleuren die met deze functie omgevormd wordt naar een array van transparantie (alpha).
- Schrijf een functie die voor een array van getallen (geheel of decimaal) zowel het kleinste als het grootste getal berekent. Schrijf nu ook de functies `min` en `max` door zo weinig mogelijk nieuwe code te schrijven.

Weather

Elke meting bevat een datum (`date`) en één meting voor temperatuur en één voor vochtigheid. Voorbeeld:
`{ date: { year: 2023, month: 1, day: 3 }, temperature: 6, humidity: 45.604 }`

Deze metingen worden voor een volledig jaar bijgehouden in één grote array (dus 365 of 366 metingen, al doet het exacte aantal metingen er eigenlijk niet toe)

Schrijf de functie `summarizePerWeek` die een array van metingen (van een volledig jaar) verwacht en een samenvatting per week teruggeeft: Je mag/moet ervan uitgaan dat de eerste meting in de array de eerste dag is van een week, en dat alle metingen opelkaar volgen en geen gaten bevatten. Je kan een samenvatting per week bereken door voor elke 7 opeenvolgende metingen de minimum en maximum meting (voor elke soort meting) in een array te stoppen. Voor een jaar aan metingen krijg je dus een array van 53 (365/7=52.14) elementen terug, één voor elke week.

De datum wordt genegeerd, maar voor elke week willen we dus voor temperatuur en vochtigheid het minimum en het maximum zien. Voorbeeld:
`[ { // first week temperature: { min: -2, max: 15 }, humidity: { min: 4.1319556493686616, max: 90.25727363205631 } }, { // second week temperature: { min: -1, max: 16 },`

Tenslotte willen we ook nog weten wat de koudste gemeten temperatuur was (min) en wat de vochtigste gemeten waarde was (max). Vul hiervoor de user-interface-functie `showInfo()` aan, zodat de juiste waarden op het scherm komen. Voor `showInfo` is er **geen** test omdat deze functie naar het scherm schrijft en geen return-value heeft.

Verder mag je voor deze oefening ook zelf gebruik maken van `generateDemoData()` om fake-test data te genereren voor één jaar. De testen gebruiken dezelfde fucntie om test data te genereren.