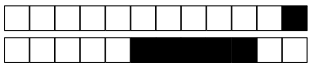


+1/1/60+

Contrôle Continu Final de LIFAP5 - Programmation fonctionnelle pour le Web 24/05/2022

Durée : 1 heure

Seule une feuille A4 recto-verso manuscrite est autorisée comme document.

Ne pas toucher aux cases situées tout en haut de la feuille, elles servent à repérer vos copies lors de la numérisation. Les questions sont ouvertes : il faut écrire la réponse dans le cadre et **ne pas noircir les cases de ces questions.** Il ne faut pas écrire en dehors des cadres, pensez à écrire la réponse au brouillon avant de la recopier.

Avant de commencer à répondre, suivez les instructions suivantes :

1. *Remplissez le cartouche de la copie-double anonymisée avec nom, prénom, numéro d'étudiant-e- et signature puis coller le rabat du cartouche (attention il ne faut **pas** coller la partie noire).*
2. *Recopier le premier numéro tout en haut du sujet sur la première page de votre copie double.*
3. *Reportez votre numéro d'anonymat imprimé sur la copie-double dans la grille ci-dessous. Si ce numéro ne comporte que 5 chiffres, mettez 0 comme premier chiffre.*

☐0	☐0	☐0	☐0	☐0	☐0
☐1	☐1	☐1	☐1	☐1	☐1
☐2	☐2	☐2	☐2	☐2	☐2
☐3	☐3	☐3	☐3	☐3	☐3
☐4	☐4	☐4	☐4	☐4	☐4
☐5	☐5	☐5	☐5	☐5	☐5
☐6	☐6	☐6	☐6	☐6	☐6
☐7	☐7	☐7	☐7	☐7	☐7
☐8	☐8	☐8	☐8	☐8	☐8
☐9	☐9	☐9	☐9	☐9	☐9

1 Transformations de tableaux

Dans cette partie, on codera des opérations de transformation de tableau en Javascript. Chaque opération va produire un nouveau tableau, donc on ne modifie **jamais** un tableau.

En plus des méthodes usuelles `map`, `filter` et `reduce` prédéfinies sur les tableaux Javascript, on considère la fonction `trier` qui va prendre en argument un tableau contenant des objets et renvoyer une nouvelle version triée de ce tableau. Pour cela, chacun des objets du tableau doit posséder un champ `cle` et ce champ doit contenir un nombre (de type `Number`). Le tableau renvoyé est trié par ordre croissant de valeur du champ `cle`.

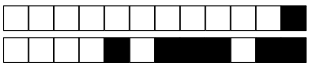
Par exemple, soit le tableau `t` suivant :

```
[{cle:5, a:"Bonjour"},{cle:2, a:"Salut", b:10}, {cle:7}]
```

alors `trier(t)` sera :

```
[{cle:2, a:"Salut", b:10}, {cle:5, a:"Bonjour"}, {cle:7}]
```

On rappelle qu'une expression n'est pas une instruction et qu'en particulier, une expression ne contient pas de déclaration de variables.



Question 1 (/1) Soit un tableau `tab` contenant des objets ayant un champ `cle`. Donner une expression Javascript permettant de récupérer un tableau contenant les éléments de `tab` dont le champ `cle` est divisible par 3. On rappelle que Javascript dispose de l'opérateur `%` qui permet de calculer le modulo (*e.g.* `7 % 5` vaut `2`). ☐f ☐j

Question 2 (/1) Soit un tableau `tab` contenant des objets ayant deux champs `v1` et `v2`. Donner une expression Javascript permettant de récupérer un tableau contenant, pour chacun des éléments de `tab`, la somme des valeurs des champs `v1` et `v2`. ☐f ☐j

Question 3 (/2) Soit un tableau `tab` contenant des nombres (de type `Number`). Donner une expression Javascript qui utilise, entre autres, la fonction `trier` et qui donne une version triée de `tab`. Remarque : le résultat doit un être un tableau de `Number`. ☐f ☐i ☐pj ☐j

Question 4 (/4) En utilisant la fontion `trier`, définir une fonction `trierGen` qui prend en argument un tableau `tab` et une fonction `calculeCle`. La fonction `calculeCle` prend en argument un élément du tableau `tab` et renvoie une valeur de clé. La fonction `trierGen` devra renvoyer un version triée du tableau `tab` par ordre croissant de la valeur de la clé calculée par `calculeCle`. Par exemple `trierGen ([{a:1,b:5},{a:2,b:3}], x => x.b)` doit renvoyer `[{a:2,b:3},{a:1,b:5}]`. ☐f ☐i ☐pj ☐j

2 λ-calcul et Javascript

Question 5 (/1) Soit M le terme de λ-calcul suivant : $\lambda x.\lambda y.(x(x\ y))$. Donner le type de M en utilisant pour y le type `number`. ☐f ☐j

Question 6 (/1) Traduire en Javascript le terme de λ-calcul suivant : $\lambda x.\lambda y.(x\ \lambda z.(y\ z))$. ☐f ☐j

Question 7 (/2) Donner un terme M' qui ne peut pas se réduire et dans lequel le terme de λ-calcul suivant se réduit : $(((((\lambda x.\lambda y.\lambda z.((x\ y)\ z))(\lambda u.\lambda w.(w\ u))))\ \lambda p.p)\ \lambda q.q)$ ☐f ☐j

Question 8 (/1) On rappelle que l'instruction `const x = expr1; return expr2` de Javascript peut être traduite en λ-calcul par $((\lambda x.M_2)M_1)$ si M_1 est la traduction de `expr1` et M_2 celle de `expr2`. Traduire en λ-calcul la fonction Javascript suivante :

```
1 function (x) {
2     const v = (y) => x(x(y));
3     return (z) => z(v);
4 }
```

☐f ☐j

Question 9 (/1) On rappelle que l'instruction `const x = expr1; return expr2` de Javascript peut être traduite en λ-calcul par $((\lambda x.M_2)M_1)$ si M_1 est la traduction de `expr1` et M_2 celle de `expr2`. Réécrire la fonction suivante sans utiliser `const` en la traduisant en λ-calcul puis en la retraduisant le résultat en Javascript.

```
function (x) {
    const a = (y) => y(x);
    return (z) => a(z)(z);
}
```

☐f ☐j

3 Javascript et programmation Web

On considère que le serveur Web qui répond à l'URL `https://exemple.com/data` envoie des documents JSON ayant la forme donnée dans la figure 1. On considère également une page Web contenant entre autres un élément `div` dont l'attribut `id` vaut `"ici"`. On suppose enfin que cette page charge le script donné dans la figure 2.

```
{
  "description": "un texte a afficher",
  "next-in": 1500
}
```

FIGURE 1 – Exemple de données envoyées par le serveur

```
const attente = (t) => setTimeout(changeAffichage, t);

const recupereDonnees = () => {
  // TODO
};

const changeAffichage = () => {
  // TODO
};

document.addEventListener("DOMContentLoaded", function () {
  attente(0);
});
```

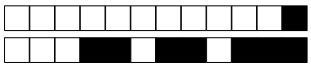
FIGURE 2 – Code Javascript

L'usage des primitives async et await est interdit dans cette partie.

Question 10 (23) Implémenter la fonction `recupereDonnees` qui déclenchera une requête GET sur l'url `https://example.com/data` et renverra une promesse de la réponse sous forme d'objet javascript obtenu à partir du json de la réponse. ☐f ☐i ☐pj ☐j

Question 11 (/3) Implémenter la fonction `changeAffichage` qui récupérera les données du serveur à l'url `https://example.com/data`, changer le contenu du `div` identifié par `"ici"` en y insérant la valeur du champ `description`, puis attendra la valeur du champ `"next-in"` millisecondes avant de redéclencher un changement d'affichage. Cette fonction devra utiliser les fonction `recupereDonnees` et `attente`. ☐f ☐i ☐pj ☐j

Question 12 (/1) Combien de fois le code de cette page va-t-il faire une requête sur `https://example.com/data` ? ☐f ☐j



+1/6/55+