+1/1/60+

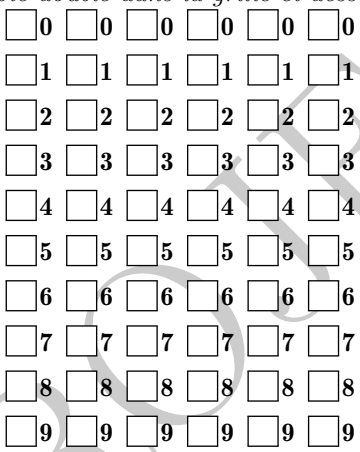
Contrôle Continu Final de LIFAP5 - Programmation fonctionnelle pour le Web **22/05/2018**

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

Ce contrôle contient un questionnaire à choix multiples qui sera corrigé automatiquement. Il est donc important de remplir avec soin les réponses au **stylo noir**. Lorsque vous choisissez une réponse, il faut noircir complètement la case correspondante. **Ne pas toucher aux cases situées tout en haut de la feuille**, elles servent à repérer vos copies lors de la correction automatisée.

Les questions faisant apparaître le symbole ♣ peuvent présenter zéro, une ou plusieurs bonnes réponses. Les questions ♢ sont ouvertes : il faut écrire la réponse dans le cadre et **ne pas noircir les cases de ces questions**. Les autres questions ont une unique bonne réponse.

Remplissez le cartouche de la copie-double anonymisée avec nom, prénom, numéro d'étudiant-e- et signature puis coller le rabat du cartouche. Reportez votre numéro d'anonymat présent sur la copie-double dans la grille ci-dessous.



1 Des promesses en `javascript` (/20)

On considère le code ci-dessous que l'on charge dans le navigateur.

```
1 function fetch_delay(url, delay=1000){
2   return new Promise((resolve, reject) => {
3       setTimeout(() => resolve(fetch(url).then(res => res.json())), delay);
4     });
5 }
```

Par ailleurs, le fichier `./ccf-fichier-A.js` contient l'objet json `{"val" : "A"}`, le fichier `./ccf-fichier-B.js` contient l'objet json `{"val" : "B"}` tandis que le fichier `./ccf-fichier-C.js` n'existe pas.

Question 1 (/2). Soit la fonction `javascript fa` définie ci-dessous :

```
function fa() {
  fetch_delay("./ccf-fichier-A.js", 500)
    .then(console.log)
    .catch(console.error);
  fetch_delay("./ccf-fichier-B.js",  1000)
    .then(console.log)
    .catch(console.error);
}
```

On exécute `fa();` au temps t_0, indiquer ce qui s'affiche dans la console :

- ☐ `Object { val: "A" }` à $t_0 + 500$ms et `Object { val: "B" }` à $t_0 + 1500$ms
- ☐ `Object { val: "A" }` à $t_0 + 500$ms et `Object { val: "B" }` à $t_0 + 1000$ms
- ☐ *Aucune des autres réponses proposées.*
- ☐ `Object { val: "B" }` à $t_0 + 500$ms et `Object { val: "A" }` à $t_0 + 1000$ms

Question 2 (/2). Soit la fonction `javascript fb` définie ci-dessous :

```
function fb() {
  fetch_delay("./ccf-fichier-A.js", 500)
    .then(console.log)
    .then(() => fetch_delay("./ccf-fichier-B.js", 1000))
    .then(console.log)
    .catch(console.error);
}
```

On exécute `fb();` au temps t_0, indiquer ce qui s'affiche dans la console :

- ☐ `Object { val: "B" }` à $t_0 + 500$ms et `Object { val: "A" }` à $t_0 + 1000$ms
- ☐ *Aucune des autres réponses proposées.*
- ☐ `Object { val: "A" }` à $t_0 + 500$ms et `Object { val: "B" }` à $t_0 + 1000$ms
- ☐ `Object { val: "A" }` à $t_0 + 500$ms et `Object { val: "B" }` à $t_0 + 1500$ms

Question 3 (/2). Soit la fonction `javascript fc` définie ci-dessous :

```
function fc() {
  fetch_delay("./ccf-fichier-C.js", 500)
    .then(console.log)
    .catch(console.error);
  fetch_delay("./ccf-fichier-B.js",  1000)
    .then(console.log)
    .catch(console.error);
}
```

On exécute `fc();` au temps t_0, indiquer ce qui s'affiche dans la console :

- ☐ `Object { val: "B" }` à $t_0 + 1500$ms
- ☐ *Une erreur* à $t_0 + 500$ms et `Object { val: "B" }` à $t_0 + 1000$ms
- ☐ `Object { val: "B" }` à $t_0 + 1000$ms
- ☐ *Une erreur* à $t_0 + 500$ms

Question 4 (/2). Soit la fonction `javascript fd` définie ci-dessous :

```
function fd() {
  fetch_delay("./ccf-fichier-C.js", 500)
    .then(console.log)
    .catch(() => fetch_delay("./ccf-fichier-B.js", 1000))
    .then(console.log)
    .catch(console.error);
}
```

On exécute `fd();` au temps t_0, indiquer ce qui s'affiche dans la console :

- ☐ *Une erreur* à $t_0 + 500$ms
- ☐ `Object { val: "B" }` à $t_0 + 1500$ms
- ☐ *Une erreur* à $t_0 + 500$ms et `Object { val: "B" }` à $t_0 + 1000$ms
- ☐ `Object { val: "B" }` à $t_0 + 1000$ms

Question 5 (/2). Soit la fonction `javascript fe` définie ci-dessous :

```
function fe() {
  fetch_delay("./ccf-fichier-C.js", 500)
    .then(console.log)
    .then(() => fetch_delay("./ccf-fichier-B.js", 1000))
    .then(console.log)
    .catch(console.error);
}
```

On exécute `fe();` au temps t_0, indiquer ce qui s'affiche dans la console :

- ☐ *Une erreur* à $t_0 + 500$ms et `Object { val: "B" }` à $t_0 + 1000$ms
- ☐ `Object { val: "B" }` à $t_0 + 500$ms
- ☐ *Une erreur* à $t_0 + 500$ms
- ☐ `Object { val: "B" }` à $t_0 + 1500$ms

Question 6 (/2). Soit la fonction `javascript va` définie ci-dessous :

```
function va() {
  let val = "Z";
  fetch_delay("./ccf-fichier-A.js", 500)
    .then(res => {val = res.val; return val;})
    .then(console.log)
    .catch(console.error);
  console.log(val);
}
```

On exécute `va();` au temps t_0, indiquer ce qui s'affiche dans la console :

- ☐ *Aucune des autres réponses proposées.*
- ☐ `"A"` à $t_0 + 0$ms et `"Z"` à $t_0 + 500$ms
- ☐ `"A"` à $t_0 + 0$ms et `"A"` à $t_0 + 500$ms
- ☐ `"Z"` à $t_0 + 0$ms et `"A"` à $t_0 + 500$ms

Question 7 (/2). Soit la fonction `javascript vb` définie ci-dessous :

```
function vb() {
  let val = "Z";
  fetch_delay("./ccf-fichier-A.js", 500)
    .then(res => res.val)
    .then(console.log)
    .catch(console.error);
  console.log(val);
}
```

On exécute `vb();` au temps t_0, indiquer ce qui s'affiche dans la console :

- ☐ `"A"` à $t_0 + 0$ms et `"Z"` à $t_0 + 500$ms
- ☐ `"A"` à $t_0 + 0$ms et `"A"` à $t_0 + 500$ms
- ☐ *Aucune des autres réponses proposées.*
- ☐ `"Z"` à $t_0 + 0$ms et `"A"` à $t_0 + 500$ms

Question 8 ♢ (/6) Dans le style le plus fonctionnel possible, écrire une fonction qui prend en paramètre une `url` supposée retourner un tableau d'entiers au format `json`. La fonction doit accéder à l'`url` et ajouter à la liste `<ul id="liste"></ul>` les entiers pairs du tableau. En cas d'erreur, un message doit être affiché sur la console.

☐f ☐i ☐pj ☐j

2 Opérations `map/reduce/filter` (/14)

On rappelle que la méthode `Array.prototype.some(p)` d'un tableau `javascript` teste si *au moins un* des éléments du tableau vérifie la propriété `p` passée en paramètre, par exemple `[1,2,3,4,5].some(x=>x%2 === 0);` produit `true` tandis que `[1,3,5].some(x=>x%2 === 0);` produit `false`. Similairement la méthode `Array.prototype.every(p)` teste si *tous les éléments* du tableau vérifient la propriété `p`.

Question 9 ♢ (*/2*) Définir une fonction `some` qui prend en paramètre un tableau `arr` et une propriété `p` et renvoie `true` si et seulement si `Array.prototype.some(p)` renvoie `true`. *Cette fonction devra utiliser `filter`.*

☐f ☐j

Question 10 ♢ (*/2*) Définir une fonction `some` qui prend en paramètre un tableau `arr` et une propriété `p` et renvoie `true` si et seulement si `Array.prototype.some(p)` renvoie `true`. *Cette fonction devra utiliser `reduce`.*

☐f ☐j

Question 11 (*/1*). Soit le tableau `tab = [1,2,3,4,5,6,7,8,9]`, donner le résultat de l'évaluation de `tab.some( n => n !== 5 && n % 5 === 0 );`

- ☐ `true`
- ☐ `false`
- ☐ *Cet appel produit une erreur à l'exécution.*

Question 12 (*/1*). Soit le tableau `tab = [1,2,3,4,5,6,7,8,9]`, donner le résultat de l'évaluation de `tab.map( n => 3 * n ).filter( n => n % 2 === 0 );`

- ☐ *Cet appel produit une erreur à l'exécution.*
- ☐ `[ 3, 9, 15, 21, 27 ]`
- ☐ `[ 3, 6, 9, 12, 15, 18, 21, 24, 27 ]`
- ☐ `[ 6, 12, 18, 24 ]`

Question 13 (*/1*). Soit le tableau `tab = [1,2,3,4,5,6,7,8,9]`, donner le résultat de l'évaluation de `tab.filter( n => n > 2 ).reduce( (acc,n)=> acc+n, 0 );`

- ☐ `9`
- ☐ *Cet appel produit une erreur à l'exécution.*
- ☐ `[42]`
- ☐ `42`

Question 14 (*/1*). Soit le tableau `tab = [1,2,3,4,5,6,7,8,9]`, donner le résultat de l'évaluation de `tab.filter( n => n <= 5 ).reduce( (acc,n)=> acc*n, 0 );`

- ☐ `0`
- ☐ *Cet appel produit une erreur à l'exécution.*
- ☐ `120`
- ☐ `[]`

Question 15 ♢ (*/2*) Que fait la fonction `let fred = (arr)=> arr.reduce((acc,x)=> acc && x, true );`

☐f ☐j

Question 16 ♢ (/2) Que fait la fonction `let fred = (arr)=> arr.reduce((acc,x)=> acc && x, false );`

☐f ☐j

Question 17 ♢ (/2) Que fait la fonction `let fcons = (arr)=> arr.reduce((acc,x)=> acc.concat(x ), []);`

☐f ☐j

3 Problème de modélisation en λ-calcul : les entiers de Church en λ-calcul (/14)

Dans le TD1, une représentation (ou codage) des booléens dans le λ-calcul avait été étudiée : après avoir codé « vrai » et « faux » par des termes du λ-calcul, on avait proposé des termes représentant les opérations « et », « ou » etc. Ce problème suit la même idée, mais il s'agit cette fois de représenter les entiers et leurs opérations. La représentation ci-après est appelée *entiers de Church*.

Pour x et f des termes du λ-calcul on écrit $f^n\ x$ pour le terme $(f(f(\ldots(f\ x)\ldots)))$ où f apparaît n fois. Formellement, par récurrence, $f^0\ x = x$ et $f^n\ x = f(f^{n-1}\ x)$. A chaque entier $n \in \mathbb{N}$ on fait correspondre le terme $\overline{n} = \lambda f.\lambda x.f^n\ x$ qui représente n l'entier dans le λ-calcul. Par exemple, $\overline{0} = \lambda f.\lambda x.x$ et $\overline{2} = \lambda f.\lambda x.f(f\ x)$. Enfin, on définit le terme $\mathbf{S} = \lambda n.\lambda f.\lambda x.f\ ((n\ f)\ x)$ pour le successeur (*i.e.* $+1$) et le terme $\mathbf{M} = \lambda n.\lambda m.\lambda f.\lambda x.(n\ (m\ f))\ x$

Question 18 (/2). Parmi, les propriétés suivantes, indiquer celle qui est correcte pour l'application de $\mathbb{N}$ dans Λ définie par $n \mapsto \overline{n}$

- ☐ *Elle est surjective.*
- ☐ *Elle est injective.*
- ☐ *Elle est bijective.*
- ☐ *Aucune des autres réponses n'est correcte.*

Question 19 ♣ (/2). Parmi les expressions suivantes, et en utilisant la définition de $\overline{n}$, indiquer celles dans lesquelles l'expression $\mathbf{S}\ \overline{3}$ se β-réduit en *une* ou en *plusieurs* étapes :

- ☐ $\lambda f.\lambda x.f(f(f(f\ \overline{0})))$
- ☐ $\lambda f.\lambda x.f(f(f(f\ x)))$
- ☐ $\overline{4}$
- ☐ *Aucune de ces réponses n'est correcte.*

Question 20 ♢ (/4) Prouver que $\mathbf{S}\ \overline{n} \overset{x}{\rightsquigarrow} \overline{n+1}$ ☐f ☐i ☐pj ☐j

Question 21 (/2). En utilisation la définition de $\overline{n}$ indiquer dans quelle expression $\mathbf{M}\ \overline{3}\ \overline{2}$ se β-réduit finalement :

☐ $\overline{2}$
☐ $\overline{5}$
☐ *Un autre terme.*
☐ $\overline{6}$

Question 22 ♢ (/2) Quelle fonction usuelle des entiers est représentée par le terme $\lambda n.\lambda m.\lambda f.\lambda x.(n\ f)\ ((m\ f)\ x)$. ☐f ☐j

Question 23 ♢ (/2) Quelle fonction usuelle des entiers est représentée par le terme $\lambda n.\lambda m.(m\ \mathbf{S})\ n$. ☐f ☐j

4 Problème de modélisation en λ-calcul : les entiers de Church en `javascript` (/14)

Question 24 ♢ (/2) Définir en `javascript` une fonction `zero` qui représente le terme $\overline{0}$. ☐f ☐j

Question 25 ♢ (/2) Définir en `javascript` une fonction `succ` qui représente le terme **S**. ☐f ☐j

Question 26 ♢ (/4) Définir en `javascript` une fonction *church(n)* qui prend en paramètre un entier n en paramètre et retourne une fonction représentant $\overline{n}$ en `javascript`. ☐f ☐i ☐pj ☐j

Question 27 ♢ (/4) Pour pouvoir afficher des entiers de Church représentés en `javascript`, on a besoin d'une fonction outil comme par exemple `let ev = n => n(x => x+1)(0);`. Justifier. ☐f ☐i ☐pj ☐j

Question 28 ♢ (/2) Évaluer `(succ(church(2)))(x => x + '.')('')`. ☐f ☐j