Exercice 1 (poids 3)

▷ Partie 1 *Algorithmes de tri*

Une liste contient, dans un ordre quelconque, les noms d'équipes nationales de football et leurs rangs dans la hiérarchie mondiale (pas d'ex-aequos). Pour remettre une telle liste dans l'ordre il existe différents algorithmes de tri. Ecrire le code de l'un d'eux, de votre choix, sans contraintes d'efficacité. Concrètement, écrire en javascript le code de la fonction *tri(x)* qui renvoie la liste ordonnée des pays sous forme de tableau. L'argument x est une variable de type tableau, de la forme :

```
<script type="text/javascript">
   var x=[  {pays:"France",rang:16},
            {pays:"Honduras",rang:32},
            {pays:"Equateur",rang:28},
            {pays:"Suisse",rang:8}]
</script>
```

Exemple: Pour cette variable x, l'instruction $alert(tri(x))$ affichera

$$["Suisse","France","Equateur","Honduras"]$$

Attention: L'utilisation de la méthode *sort()* n'est pas autorisée.

▷ Partie 2 *Expressions régulières*

Ecrire l'expression régulière qui permet de valider le résultat d'un match, c'est-à-dire qui accepte les expressions de la forme "*texte* - *texte* : *nb* - *nb*", comme par exemple "Suisse - Espagne : 1 - 0".
Pour simplifier on choisit les contraintes suivantes :

- *texte* commence par une majuscule, n'en a qu'une, et est en un seul mot d'au maximum 20 caractères
- *nb* n'est pas supérieur à 5.

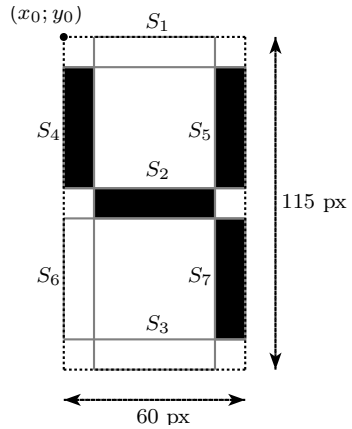
▷ Partie 3 *HTML-Javascript*

Les noms de diverses équipes figurent dans le tableau nommé *equipes*. On souhaite générer aléatoirement un match entre deux des équipes du tableau, et son résultat. Un texte de la forme "Suisse - Espagne : 1 - 0" doit finalement être affiché dans une *div*. Il est impératif que le choix du match soit tel que toutes les confrontations possibles aient la même probabilité d'être obtenues. De plus, le résultat du match s'obtiendra après le tirage d'un nombre aléatoire, en lisant les deux premières décimales voisines inférieures à 6. Le nombre aléatoire 0.2694**25**83... donnerait donc le score **2-5**. On suppose que le nombre aléatoire permet toujours d'obtenir un score.
Compléter le code suivant de sorte que lors du clic sur un bouton, à créer, le match et le résultat soient générés et que l'affichage du texte dans la *div* se fasse.

```
<script type="text/javascript">
   var equipes=["Espagne","Suisse","Honduras","Chili"]
   //... à compléter
</script>
<body>
   <div id="match"></div>
   //... à compléter
</body>
```

Exercice 2 (poids 3) *Affichage digital*

L'affichage digital d'un chiffre c s'obtient par sept segments rectangulaires $S_1, S_2, ..., S_7$ dont l'état peut être allumé (état=1) ou éteint (état=0), comme illustré ci-contre.
Pour écrire les 10 chiffres (0 à 9) en base 2, on utilise 4 bits. Ces bits $E_3E_2E_1E_0$ sont les entrées du circuit logique avec lequel l'affichage digital sera réalisé. Les 4 bits utilisés permettent aussi de coder les nombres de 11 à 15, dont on n'a pas besoin pour ce problème.



a) Voici la table de vérité complétée pour les chiffres c de 0 à 5. Compléter les 4 dernières lignes.

c	E_3	E_2	E_1	E_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7
0	0	0	0	0	1	0	1	1	1	1	1
1	0	0	0	1	0	0	0	0	1	0	1
2	0	0	1	0	1	1	1	0	1	1	0
3	0	0	1	1	1	1	1	0	1	0	1
4	0	1	0	0	0	1	0	1	1	0	1
5	0	1	0	1	1	1	1	1	0	0	1
6											
7											
8											
9											

b)
- On considère le segment S_4. Compléter le tableau de Karnaugh ci-contre pour S_4. Ne compléter que les 9 cases correspondant aux "entrées" c de 0 à 9.
- Compléter cette table en plaçant des 1 dans les cases non remplies, c'est-à-dire correspondant à c entre 10 et 15. Finalement écrire l'expression logique la plus simple de S_4.

E_3E_2 \ E_1E_0	00	01	11	10
00				
01				
11				
10				

c) Comme illustré, les chiffres sont affichés dans des zones rectangulaires de taille 60x115px. Nous utilisons sept constructeurs, un par segment : $S1(x_0, y_0)$, $S2(x_0, y_0)$, ..., $S7(x_0, y_0)$. Ecrire le code du constructeur $S2(x_0, y_0)$. Ce constructeur a les attributs x_0 et y_0 qui sont les coordonnées du point en haut à gauche de la zone rectangulaire (voir dessin), et une méthode $print()$ qui dessine le segment, un rectangle noir de taille 50x5px. Le contexte du canvas est ctx.

d) Ecrire le code de la fonction $f4(x_0, y_0)$ qui dessine le chiffre 4, identique à celui représenté au début de cet exercice. Utiliser pour cela les constructeurs $S1, ..., S7$. Ne pas afficher les segments éteints.

e) Ecrire le code de la fonction $compteRebours(debut)$. Lorsque cette fonction est appelée, un compte à rebours sur deux digits doit démarrer et égrainer les secondes à partir de *debut* (un entier non nul inférieur à 100 dont on ne demande pas de vérifier la validité). Par exemple, $compteRebours(12)$ affichera successivement 12, 11, 10, 09 , 08 , ... , 00. Le canvas a la dimension 125x115px pour permettre l'affichage de 2 chiffres.
Pour programmer cette fonction, vous disposez de deux fonctions (ne pas en écrire le code !)

- La fonction $efface()$, qui efface votre canvas.
- La fonction $affiche(c, x_0, y_0)$ qui permet d'afficher le chiffre c au format digital en position $(x_0; y_0)$.

Exercice 3 (poids 2) *Le chiffrement de César*

Le chiffrement de César est une ancienne méthode, utilisée par Jules César, pour coder un message secret. La méthode consiste à décaler, toujours du même côté, les lettres de l'alphabet. Ce décalage est fixe et circulaire : lorsqu'on se trouve à la fin de l'alphabet, on recommence au début ; autrement dit : "A suit Z".

Exemple : Si on choisit $decalage = 3$, on codera les messages à l'aide du tableau ci-dessous :

Non codé	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Codé	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C

Ecrire la fonction $cesar(message, decalage)$ qui renvoie en majuscule le message codé. L'argument *message* est une chaîne de caractères contenant uniquement des espaces et des lettres (majuscules et minuscules, sans accents). L'argument *decalage* est un entier entre 1 et 25 qui indique de combien de lettres on veut décaler l'alphabet. Il n'est pas demandé de tester la validité des arguments saisis.

Exemple : $alert(cesar('Salut', 3))$ doit afficher dans une boîte de dialogue le message $'VDOXW'$.

Exercice 4 (poids 2) *Bases de données*

TafService est une agence de travail intérimaire. Des bases de données sont utilisées pour la gestion des missions qui lui sont confiées. Pour illustrer de manière simplifiée, on considère les tables *employes*, *competences* et *missions*. Elles contiennent des informations qui permettent la planification des prochaines missions.

employes (e)
ide (primary key)
nom
region
natel

competences (c)
idc (primary key)
ide
job
qual

missions (m)
idm (primary key)
job
qual
region

TafService propose ses services dans deux régions (1 ou 2). Les employés ont précisé leur région en indiquant "1" ou "2". Un employé qui accepte de travailler dans les deux régions aura inscrit "3".
Un employé peut avoir des compétences dans divers jobs, avec des qualifications variables. Les jobs sont codés "A","B","C". Les qualifications sont des valeurs supérieures ou égales à 1.
Les clients qui proposent des missions indiquent la région (1 ou 2), le job (A,B ou C) et la qualification minimale nécessaire que l'employé doit avoir.

a) Quel(s) employé(s) pourrai(en)t être engagé(s) pour la mission no. 1 ? Répondre à partir des enregistrements des tables de l'annexe 1.

Indiquer les instructions (requêtes) SQL qui permettent de répondre aux demandes suivantes.

b) Afficher les noms, régions et natel des employés, par ordre décroissant de régions et croissant de noms.

c) Pour quel(s) job(s) *TafService* peut-il proposer l'employé de meilleure qualification ? Ecrire la requête qui permet de répondre à cette question.

d) Afficher les noms des employés ayant des qualifications supérieures à 1 pour le job "B" et pouvant être engagés dans la région 2.

e) On considérera que la *qualification globale* est la somme des qualifications. Afficher les noms des employés et leur *qualification globale*, le plus qualifié d'abord.

f) Afficher le nom des employés pouvant être engagés pour la mission no. 11.

g) L'employé "Georges" a amélioré de 1 toutes ses qualifications en suivant une formation. Mettre à jour la table concernée.

Annexe 1 — *Tables **partielles** pour les bases de données, exercice 4*

Table employes

ide	*nom*	*region*	*natel*
1	Bernie	1	079 111 11 11
2	Charlie	2	076 123 45 67
3	Ronnie	3	077 987 65 43
4	Andy	3	078 789 01 23
5	Mike	2	079 246 80 24
...	...	...	...

Tables competences

idc	*ide*	*job*	*qual*
1	1	A	1
2	1	B	3
3	2	C	3
4	3	B	2
5	3	C	2
6	4	A	3
7	4	C	2
8	5	A	2
9	5	B	1
10	5	C	3
...	...	...	...

Table missions

idm	*job*	*qual*	*region*
1	A	2	2
2	A	2	1
3	B	1	1
4	B	2	1
5	C	2	2
...	...	...	...

Annexe 2 *Expressions régulières*

Expression	*Description*
$[abc]$	attend un caractère parmi ceux proposés entre crochets
$[\hat{}abc]$	attend un caractère différent de ceux proposés entre crochets
$[0-9]$	attend un chiffre
$[A-Z]$	attend une lettre majuscule
$[a-z]$	attend une lettre minuscule
$[A-z]$	attend une lettre sans distinction de casse
$(rouge\|bleu\|vert)$	attend une chaîne parmi les chaînes proposées
$X+$	accepte toute chaîne contenant au moins 1 occurrence de X
$X*$	accepte toute chaîne contenant 0 ou plus occurrence(s) de X
$X?$	accepte toute chaîne contenant 0 ou 1 occurrence de X
$X\$$	accepte toute chaîne se terminant par X
$\hat{}X$	accepte toute chaîne commençant par X
$X\{n\}$	le caractère X est répété n fois
$X\{n,m\}$	le caractère X est répété entre n et m fois
$X\{n,\}$	le caractère X est répété n fois ou plus
$\backslash d$	identique à $[0-9]$
$\backslash D$	identique à $[\hat{}0-9]$
$\backslash w$	indique un caractère alphanumérique
$\backslash s$	indique un espace blanc
.	indique n'importe quel caractère. Il autorise donc tout

Annexe 3 *Quelques méthodes de l'objet Math*

Expression	*Description*
abs(x)	renvoie la valeur absolue de x
ceil(x)	renvoie l'entier supérieur à x
exp(x)	renvoie e^x
floor(x)	renvoie l'entier inférieur à x
max(x,y,z,...,n) et *min(x,y,z,...,n)*	renvoient le maximum (minimum) des nombres entrés en paramètres
pow(x,y)	renvoie x^y
random()	renvoie un nombre aléatoire dans l'intervalle $[0;1[$
round(x)	renvoie l'entier le plus proche de x
sqrt(x)	renvoie la racine carrée de x

Annexe 4 — *Quelques méthodes de l'objet Array*

Expression	*Description*
x.concat(tab1, tab2[, tab3, ...])	Cette méthode concatène x avec d'autres tableaux. x est modifié
x.pop()	Cette méthode supprime le dernier élément du tableau et retourne sa valeur
x.push(valeur1[, valeur2, ...])	Cette méthode ajoute un ou plusieurs éléments à la fin du tableau x. Elle retourne le nombre d'éléments du tableau modifié
x.unshift(valeur1[,valeur2, ...])	Cette méthode ajoute un ou plusieurs éléments au début du tableau x. Elle retourne le nombre d'éléments du tableau modifié
x.reverse()	Cette méthode inverse l'ordre des éléments du tableau
x.sort()	Cette méthode trie les éléments du tableau
x.shift()	Cette méthode supprime le premier élément du tableau et retourne sa valeur
x.slice(indiceDebut,indiceFin)	Cette méthode renvoie le sous-tableau contenant les éléments de *indiceDebut* (inclus) à *indiceFin* (non inclus). Le tableau initial x n'est pas modifié

Annexe 5 — *Quelques méthodes de l'objet String*

Expression	*Description*
x.charAt(index)	Lecture d'un caractère
x.charCodeAt(index)	Code Unicode d'un caractère ("A":65, "a":97)
String.fromCharCode(x)	Caractère dont l'Unicode est le nombre x
x.toUpperCase()	Casse : tout en majuscule
x.toLowerCase()	Casse : tout en minuscule
x.toString()	Conversion du nombre x en String
x.split(car)	Conversion String à Array
tab.join(car)	Conversion Array à String
txt1.concat(txt2,txt3,...)	Concaténation
x.replace(searchvalue,newvalue)	Remplacement
x.slice(start,end)	Renvoie la sous-chaîne d'indice *start* (inclus) à *end* (non inclus)
x.slice(start)	Renvoie la sous-chaîne d'indice *start* (inclus) à la fin
x.slice(-n)	renvoie la sous-chaîne contenant les n derniers caractères

Annexe 6 — *Syntaxe et fonctions SQL*

Expression	*Syntaxe*
AND/OR	SELECT ... FROM ... WHERE ... AND/OR ...
AS (alias)	SELECT... AS... FROM...
BETWEEN	SELECT ... FROM ... WHERE ... BETWEEN ... AND...
DELETE	DELETE FROM ... WHERE...
DROP TABLE	DROP TABLE...
GROUP BY	SELECT ... FROM ... WHERE ... GROUP BY...
IN	SELECT ... FROM ... WHERE ... IN...
INSERT INTO	INSERT INTO ... VALUES ...
LIKE	SELECT ... FROM ... WHERE ... LIKE ...
ORDER BY	SELECT ... FROM ... ORDER BY ... [ASC/DESC]
SELECT	SELECT ... FROM ...
SELECT *	SELECT * FROM ...
SELECT DISTINCT	SELECT DISTINCT ... FROM ...
UPDATE	UPDATE ... SET ... WHERE ...
WHERE	SELECT ... FROM ... WHERE ...

Expression	*Description*
AVG()	moyenne
COUNT()	nombre d'enregistrements
MAX()/MIN()	plus grande/petite valeur
SUM()	somme
LEN()	longueur d'un champ texte
ROUND()	arrondi d'une valeur numérique au nombre de décimales éventuellement spécifié
CONCAT()	concaténation

Exercice 1

▷ **Partie 1**

```
var x=[{pays:"France",rang:16},{pays:"Honduras",rang:32},{pays:"
   Equateur",rang:28},{pays:"Suisse",rang:8}]
var xtri=[]
do {                          //tri par selection
    var r=x[0].rang
    var kr=0
    for (var k=0;k<x.length;k++){
        if (x[k].rang<r) {
           r=x[k].rang
           kr=k
        }
    }
    xtri.push(x[kr].pays)
    for (var m=kr;m<x.length;m++){   //enlever element xk de x
    x[m]=x[m+1]
    }
    x.pop()
}
while (x.length>0)
alert(xtri)
```

▷ **Partie 2**

```
var reg=/^[A-Z][a-z]{0,19}\s-\s[A-Z][a-z]{0,19}\s:\s[0-5]\s-\s[0-5]$/
```

▷ **Partie 3**

```
<script>
    var equipes=["Suisse","Honduras","Equateur","France"]
    var n=equipes.length
    function f(){
    var a=parseInt(Math.random()*n)
    do {
        var b=parseInt(Math.random()*n)
    } while (a==b)
    var c=Math.random()
    c=c.toString()
    var k=1
    do {
      k=k+1
    } while (c[k]>5 || c[k+1]>5)
    var texte=equipes[a]+" - "+equipes[b]+" : " + c[k] + " - " + c[k+1]
    document.getElementById("match").innerHTML=texte }
</script>
<body>
<div id="match"></div>
<button onclick="f()"> GENERE MATCH </button>
</body>
```

Exercice 2

a) Table de vérité

c	E_3	E_2	E_1	E_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7
0	0	0	0	0	1	0	1	1	1	1	1
1	0	0	0	1	0	0	0	0	1	0	1
2	0	0	1	0	1	1	1	0	1	1	0
3	0	0	1	1	1	1	1	0	1	0	1
4	0	1	0	0	0	1	0	1	1	0	1
5	0	1	0	1	1	1	1	1	0	0	1
6	0	1	1	0	1	1	1	1	0	1	1
7	0	1	1	1	1	0	0	0	1	0	1
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	1	1	0	1

b) Expression logique: $E_3 + \overline{E_3E_2E_1E_0} + \overline{E_3}E_2\overline{E_1} + \overline{E_3}E_2E_1\overline{E_0}$

E_3E_2 \ E_1E_0	00	01	11	10
00	1	0	0	0
01	1	1	0	1
11	*1*	*1*	*1*	*1*
10	1	1	*1*	*1*

c)
```
function S2(_x0,_y0){
    this.x0=_x0
    this.y0=_y0
    this.print=function(){
        ctx.fillRect(this.x0+5,this.y0+55,50,5)
    }
}
```

d)
```
function f4(x0,y0){
    new S2(x0,y0).print()
    new S4(x0,y0).print()
    new S5(x0,y0).print()
    new S7(x0,y0).print()
}
```

e)
```
function compteRebours(debut){
if (debut>=0){
    efface()
    unit=debut
    dec=Math.floor(debut/10)
    affiche(unit,65,0)
    affiche(dec,0,0)
    var t=setTimeout(function (){compteRebours(debut-1)},1000)
}
}
```

Exercice 3

```
function maj(string){
    return string.toUpperCase().split("")
}
function decale(tab,k){
    var newTab=[]
    for(var i=0; i< tab.length;i++){
        newTab[i]=tab[(i+k)%tab.length]
    }
    return newTab
}
function chiffrementCesar(string,decalage){
    var alphabet='ABCDEFGHIJKLMNOPQRSTUVWXYZ'
    var tAlphabet=alphabet.split("")
    var tAlphabetCode=decale(tAlphabet, decalage)
    var tab=maj(string)
    var newTab=[]
    for (var i=0;i<tab.length;i++){
        for( var j=0; j<26;j++){
            if(tab[i]==tAlphabet[j]){
                newTab[i]=tAlphabetCode[j]
            }
        }
    }
    return newTab.join("")
}

alert(chiffrementCesar('Salut',3))
```

Autre version

```
function cesar(message,decalage){
    var out=[]
    message=message.toUpperCase()
    for (k=0;k<message.length;k++){
        if (message[k]!=" "){
        var x=message.charCodeAt(k)+parseInt(decalage)
        if (x>90) {x=x-26}
        out.push(String.fromCharCode(x))
    }
    else {out.push(" ")}
    }
    return(out.join(""))
}
```

Exercice 4

1. Mike et Andy
2. select nom, region, natel from employes order by region desc, nom;
3. select distinct job from competences where qual=(select max(qual) from competences);
4. select nom from employes, competences where employes.ide=competences.idc and job="B" and qual>1 and region>=2;
5. select sum(qual) as qualif, nom from competences, employes where employes.ide = competences.ide group by nom order by qualif desc;
6. select distinct nom from employes, competences, missions where employes.ide=competences.ide and competences.job=missions.job and (missions.region=employes.region or employes.region=3) and competences.qual>=missions.qual and idm=11;
7. update competences set qual=qual+1 where ide=(select ide from employes where nom="Georges") ;