Exercice 1 (poids 2)

Codage, tests et expressions régulières

a. Compléter le programme ci-contre qui permet de convertir un nombre **binaire en un nombre entier** positif. L'utilisateur saisit un nombre dans une balise *input*. La conversion s'exécutera lors d'un clic sur un bouton. Faire afficher le nombre converti dans une fenêtre *alert*.
Dans cette première partie, on ne demande pas de tester si l'utilisateur a effectivement saisi une valeur plausible.

```
<script type="text/javascript">
   //... à compléter
</script>
<body>
   // ... à compléter
</body>
```

b. On s'intéresse maintenant au test de la chaîne de caractères entrée par l'utilisateur. Il s'agit de vérifier que le nombre saisi soit bien un nombre en base 2. Pour cela on crée une fonction qui retournera *true* si l'expression est un nombre exprimé en base 2, et *false* sinon.

Par exemple: *true* pour "11010" et "0" et *false* pour "0100" et "10014" et "101ab10"

Note: On fait le choix de refuser le nombres dont le bit de gauche vaut 0.

On vous demande deux versions de ce test.

- **Version 1 :** Ecrire le code de la fonction *test(x)*. Cette fonction n'utilise pas les expressions régulières. L'argument x est la chaîne de caractères à tester.
- **Version 2 :** Compléter le code de la fonction *testReg(x)*, qui utilise les expressions régulières. Vous trouvez dans *l' annexe 1* un rappel sur les expressions régulières.

```
function testReg(x){
   var resultat=false
   var regExp=//... à compléter
   if (regExp.test(x)){
      resultat=true
   }
   return resultat
}
```

Exercice 2 (poids 2)

Bases de données

Une compagnie aérienne utilise une base de données pour mémoriser des informations sur ses pilotes, sur les trajets qu'elle propose à partir de Genève, ainsi que sur les engagements de ses pilotes sur des vols. Les tables partielles, qui constituent cette base, se trouvent dans *l'annexe 2* du document.

a. Utiliser les enregistrements des tables données dans *l' annexe 2* pour indiquer le résultat de la requête :

select min(distance) from trajets where escale=true or prix>200

b. Ecrire le code SQL des requêtes qui permettent d'effectuer les actions suivantes:

- Ajouter dans la table des pilotes "Alain P., 25 ans". La clé primaire est *piloteID*; elle s'incrémente automatiquement.
- Mettre à jour le trajet dont le champ *trajetID* vaut 18. Dorénavant, ce trajet a une escale et la distance devient 1030 kilomètres.
- Afficher le nom et la distance de la destination la plus éloignée.
- Un pilote de la compagnie s'appelle "Bob L.". Afficher la liste des destinations sur lesquelles "Bob L." a volé.
- Afficher l'âge moyen des pilotes qui ont volé en 2011.

c. Ecrire le code de la requête qui a produit, pour les enregistrements de *l' annexe 2*, le résultat ci-contre.

Indications :

- La colonne *prix100km* est un champ calculé qui correspond au prix pour 100 kilomètres.
- La colonne *arrivée* est obtenue par concaténation de champs et de chaînes de caractères. L'instruction est *concat(a,b)*. Les 0 et 1 correspondent directement à *false* et *true* du champ escale.

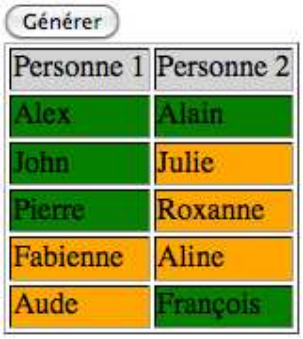
prix100km	*arrivée*
23.6729	Rome(escale 0)
23.4834	Madrid(escale 0)
22.6415	Madrid(escale 1)
21.327	Paris(escale 0)
17.3218	Lisbonne(escale 0)
13.553	Berlin(escale 0)
13.125	Lisbonne(escale 1)

Exercice 3 (poids 3) *Programmation dynamique*

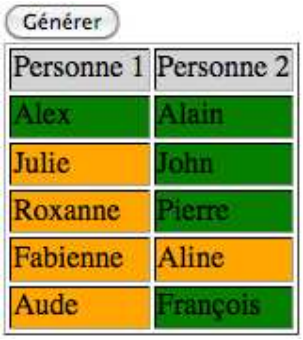
a. Compléter le programme ci-dessous afin que lorsque l'utilisateur presse le bouton "Générer", le contenu du tableau p apparaisse dynamiquement sous la forme ci-contre. Le programme doit afficher dans une *div* une table de 2 colonnes dont les cellules contiennent les noms listés dans le tableau p, sans en changer l'ordre. La couleur de fond des cellules doit être différente selon le type ("g"/"f") de la personne.


```
<script type="text/javascript">
var p=[{type:"f", nom:"Aline"},{type:"f", nom:"Aude"},
       {type:"g", nom:"François"},{type:"g", nom:"Alain"},
       {type:"f", nom:"Julie"},{type:"f", nom:"Roxanne"},
       {type:"g", nom:"Pierre"},{type:"g", nom:"John"},
       {type:"g", nom:"Alex"},{type:"f", nom:"Fabienne"}]
function generer(x){
   // ... à compléter
   }
</script>
<body>
   <button onclick='generer(p)'>Générer</button>
   <div id="affiche"></div>
</body>
```

b. Ecrire le code de la fonction $alea(x)$ qui renvoie un tableau aléatoire obtenu en permutant les éléments du tableau x donné en argument. Dans notre exemple, l'instruction $generer(alea(p))$ affichera le tableau aléatoire comme illustré ci-contre. Cette table contient des couples aléatoirement formés à partir des éléments du tableau p.

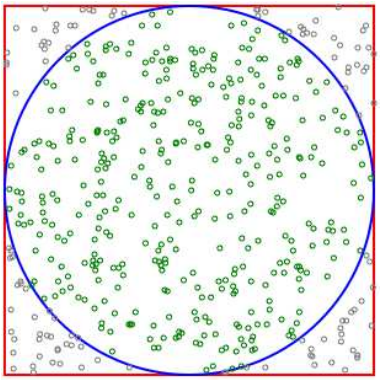


c. On souhaite que la fille soit mentionnée en premier lorsque le couple est mixte.
Ainsi si sur une ligne une fille et un garçon apparaissent, on souhaite toujours afficher le prénom féminin à gauche.
Si le tableau p qu'on utilise ne respecte pas cette nouvelle contrainte, on doit donc le modifier avant de créer dynamiquement la table.
Ecrire le code de la fonction *honneurAuxFilles(x)* qui renvoie un tableau satisfaisant cette contrainte, à partir de x un tableau donné en argument qui contient un nombre pair d'éléments.



L'instruction *generer(honneurAuxFilles(alea(p)))* permet alors d'afficher un tableau aléatoire avec les filles dans la première colonne lorsque les couples sont mixtes, comme illustré ci-dessus.

Exercice 4 (poids 3) *CSS, timers, canvas et programmation objets*



La méthode de **Monte-Carlo** a pour but d'estimer des valeurs numériques en utilisant des procédés aléatoires. **Nous allons utiliser ce procédé pour approcher la valeur de π.**
Considérons une cible, c'est-à-dire un carré rouge de côté donné et le cercle bleu qui lui est tangent intérieurement, comme illustré ci-contre. Des flèches sont lancées de manière aléatoire contre la cible. La proportion des flèches à l'intérieur du cercle par rapport au nombre total de flèches lancées donne une estimation de $\frac{\pi}{4}$.
Cette estimation est théoriquement d'autant meilleure que le nombre de flèches tirées est grand.

a. Expliquer pourquoi la proportion des flèches à l'intérieur du cercle par rapport au nombre total de flèches lancées donne une estimation de $\frac{\pi}{4}$.

b. Ecrire le code du constructeur *Simulation* dont les attributs et les méthodes sont :

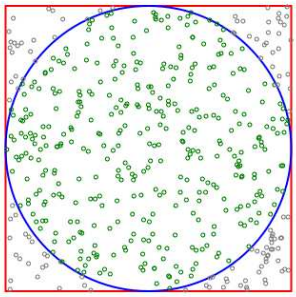
- **Un attribut** *valeurs*. Il s'agit d'un tableau qui contient la position des flèches. Par exemple, le tableau $[\{px : 200, py : 300\}, \{px : 100, py : 60\}]$ contient deux flèches aux positions $(200, 300)$ et $(100, 60)$. Lors de la création d'un objet de type *Simulation*, ce tableau est vide. Le contenu de ce tableau est mis à jour par la méthode $fleches()$.

Diagramme de la classe **Simulation**
valeurs: array *cote*: int *nbFleches*: int
$fleches()$: void $dessinerCible()$: void $dessinerFleches()$: void $estDans(x, y)$: boolean $estimation()$: double

- **Un attribut** *cote* qui représente la mesure du côté de la cible.
- **Un attribut** *nbFleches* qui représente le nombre de flèches lancées.
- **La méthode** $fleches()$ met à jour l'attribut *valeurs*. Il y ajoute *nbFleches* flèches de positions aléatoires variant entre 0 et *cote*, donc dans le carré.
- **La méthode** $dessinerCible()$ dessine la cible (cercle bleu et carre rouge) en tenant compte de l'attribut *cote*.
- **La méthode** $dessinerFleches()$ dessine des flèches sur la cible. Il s'agit de cercles verts ou gris selon qu'ils sont intérieurs ou extérieurs au cercle bleu de la cible. Le rayon de ces cercles vaut 2 *px*.
- **La méthode** $estDans(x, y)$ retourne *true* si le point (x, y) est dans le cercle bleu, *false* sinon.
- **La méthode** $estimation()$ retourne une estimation de π, avec 2 chiffres après la virgule.

Nous voulons réaliser le programme suivant: on crée une simulation en dessinant une cible avec ses flèches. On calcule alors l'estimation de π pour cette simulation. On stocke cette estimation dans un tableau que l'on appelle *listeEstimations*. Toutes les secondes, on rafraîchit l'image en créant une nouvelle simulation. On calcule alors la nouvelle estimation de π et on met à jour la valeur moyenne de toutes les estimations effectuées. Une capture du programme se trouve ci-dessous. Quant au corps du programme, il se trouve ci-contre.

Estimation:3.07
Moyenne:3.152



```
<style type="text/css">
      // ... définition du CSS des div
</style>
<script type="text/javascript">
   var ctx,c
   var listeEstimations=[]
   function Simulation(nombre,cote){
      // ... constructeur Simulation
                }
   function moyenne(){
      // ... calcul de la valeur moyenne
      }
   function init(){
      c = document.getElementById('canvas');
      ctx = canvas.getContext('2d');

      c.width=document.body.clientWidth;
      c.height=document.body.clientHeight;
      }
   function lancer(nombre,taille){
      // ... initialisation, création de l'
         objet Simulation, dessin et affichage
         d'une estimation et de la valeur
         moyenne
      }
</script>
<body onload="
         var n=prompt('Taille de la cible');
         var nb=prompt('Nombre de flèches');
         ...">
   <div id="estim"></div>
   <div id="moyenne"></div>
   <canvas  id="canvas"
         style="border:none"></canvas>
</body>
```

c. Compléter le code de l'événement *onload* de la balise *body* afin que le programme lance une simulation toutes les secondes. Utiliser la fonction *lancer(nombre,taille)*.

d. Compléter le code de la fonction *moyenne()*. Cette fonction récupère les estimations de π enregistrées dans le tableau *listeEstimations* et retourne la valeur moyenne arrondie à 3 chiffres après la virgule.

e. Compléter le code de la fonction *lancer(nombre,taille)*. Cette fonction est appelée toutes les secondes. Elle initialise la page, crée un objet *Simulation*, affiche la cible et les flèches. Elle affiche également la valeur moyenne des estimations de π.

f. Compléter la balise CSS afin de redéfinir le format des *div*. Dans ce programme, toutes les *div* ont en effet un texte de couleur brune, de police arial et de taille 3 fois plus grande que la taille standard.

Annexe 1

Rappel sur les 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.
$n+$	accepte toute chaîne contenant au moins 1 occurrence de n.
$n*$	accepte toute chaîne contenant 0 ou plus occurrence(s) de n.
$n?$	accepte toute chaîne contenant 0 ou 1 occurrence de n.
$n\$$	accepte toute chaîne se terminant par n.
$\hat{}n$	accepte toute chaîne commençant par n.

Annexe 2

Tables partielles pour les bases de données

Table pilotes

piloteID	*nom*	*age*
1	Alexandre T.	26
2	Charles H.	54
3	Marie Z.	41
4	Hubert F.	37
...	...	...

Table vols

volID	*piloteID*	*trajetID*	*date*
1	2	2	2012-05-01
2	1	2	2011-10-04
3	3	1	2011-03-27
4	2	5	2012-01-09
5	4	4	2010-12-14
6	2	2	2012-01-08
...	...	...	...

Table trajets

trajetID	*destination*	*escale*	*distance*	*prix*
1	Berlin	false	868	135
2	Paris	false	422	90
3	Lisbonne	true	1600	210
4	Rome	false	697	165
5	Lisbonne	false	1501	260
6	Madrid	false	1022	240
7	Madrid	true	1060	240
...	...	...	...	...

Exercice 1

a.
```
<script type="text/javascript">
   function convertir(){
      var out=0, j=0
      var nb=document.getElementById('i1').value
      for (  var i= nb.length-1; i > -1; i-- ) {
         out+=parseInt(nb[i])*Math.pow(2,j)
         j++
      }
      alert(out)
   }
</script>
<body>
   <input id='i1'/>
   <button onclick='convertir()'>Convertir</button>
</body>
```

b.
- ```
  function test(x){
     if (x=="0") {return true}
     else{
        var ok=true
        for (var i=0;i<x.length;i++){
           if (!(x[i]==0 || x[i]==1)) {ok=false}
        }
        if (x[0]==0){ok=false}
        return ok
     }
  }
  ```
- ```
  function testReg(x){
     var resultat=false
     var regExp=/^(1[0-1]*|0)$/
     if (regExp.test(x)){
        resultat=true
     }
      return resultat
  }
  ```

Exercice 2

a. 1022

b.
- ***insert into*** *pilotes*
  values*(0,'Alain P.',25);*
- ***update*** *trajets*
  set *escale=true, distance=1030* ***where*** *trajetID=18;*
- ***select*** *destination, km* ***from*** *trajets*
  where *distance=(****select max****(distance)* ***from*** *trajets);*

- ***select distinct*** *destination* ***from*** *vols, trajets, pilotes*
  where *nom="Bob L."*
  and *pilotes.piloteID=vols.piloteID*
  and *vols.trajetID=trajets.trajetID*
  order by *destination;*
- ***select avg(***age***)*** ***from*** *pilotes, vols*
  where year(date***)***=2011
  and *pilotes.piloteID=vols.piloteID;*

c. ***select concat(***destination,"(escale ",escale,")"***)*** ***as*** *arrivée,*
*prix/dist*100* ***as*** *prix100km* ***from*** *trajets*
order by *prix100km* ***desc;***

Exercice 3

a.
```
function generer(x){
   for ( var i = 0; i < x.length; i++ ) {
      if(x[i].type=='f'){
         x[i].couleur="orange"
      }
      else{
         x[i].couleur="green"
      }
   }
   var aux="<table border='1'><tr bgcolor='lightgrey'><td>Personne 1</td>"
   aux+="<td>Personne 2</td></tr>"
   for ( var i = 0; i < x.length; i=i+2 ) {
      aux+="<tr><td bgcolor="+x[i].couleur+">"
      aux+=x[i].nom+"</td><td bgcolor="+x[i+1].couleur+">"
      aux+=x[i+1].nom+"</td></tr>"
   }
   document.getElementById('affiche').innerHTML=aux+"</table>"
}
```

b.
```
function choisirUnElement(utilise){
   var nb, trouve=false
   while(!trouve){
      nb=Math.floor(Math.random()*utilise.length)
      if (utilise[nb]==0){
         trouve=true
         utilise[nb]=1
      }
   }
   return nb
}
```

```
function alea(x){
   var nouveauTableau=[],utilise=[]
   for ( var i = 0; i < x.length; i++ ) {
      utilise[i]=0
   }
   for ( var j = 0; j < x.length; j++ ) {
      nouveauTableau[j]=x[choisirUnElement(utilise)]
   }
   return nouveauTableau
}
```

c.
```
function honneurAuxFilles(x){
   var temp
   for ( var i = 0; i < x.length; i+=2) {
      if ((x[i].type=="g")&&(x[i+1].type=="f")){
         temp=x[i+1]
         x[i+1]=x[i]
         x[i]=temp
      }
   }
   return x
}
```

Exercice 4

a. $E = \dfrac{\pi\left(\frac{c}{2}\right)^2}{c^2} = \dfrac{\pi c^2}{4 \cdot c^2} = \dfrac{\pi}{4}$

b.
```
function Simulation(nombre,cote){
   this.cote=cote || 250
   this.nbFleches=nombre || 500
   this.valeurs=[]

   this.dessinerCible=function(){
      ctx.beginPath()
      ctx.strokeStyle = "red"
      ctx.lineWidth = 2
      ctx.moveTo(50,50)
      ctx.lineTo(50 + this.cote,50)
      ctx.lineTo(50+this.cote,50+this.cote)
      ctx.lineTo(50,50 + this.cote)
      ctx.closePath();
      ctx.stroke();

      ctx.beginPath()
      ctx.strokeStyle = "blue"
      ctx.lineWidth = 2
      ctx.arc(50+this.cote*0.5,50+ this.cote*0.5,this.cote*0.5,0,2*Math.PI,
         true)
      ctx.closePath();
      ctx.stroke();
   }
```

```
    this.fleches=function(){
        var posX,posY
        for ( i = 0; i < this.nbFleches; i++ ) {
            posX=50+ Math.floor(Math.random()*this.cote)
            posY=50+ Math.floor(Math.random()*this.cote)

            this.valeurs.push({x:posX,y:posY})
        }
    }

    this.estDans=function(px,py){
        var res=false
        if(Math.sqrt(Math.pow((px-(50+this.cote*0.5)),2)+Math.pow((py-(50+this.
            cote*0.5)),2))<=this.cote*0.5){
            res=true
        }
        return res
    }

    this.dessinerFleches=function(){
        var couleur
        for ( i = 0; i <this.valeurs.length; i++ ) {
            if(this.estDans(this.valeurs[i].x,this.valeurs[i].y)){
                couleur="green"
            }
            else{
                couleur="grey"
            }
            ctx.beginPath()
            ctx.strokeStyle = couleur
            ctx.lineWidth = 1
            ctx.arc(this.valeurs[i].x,this.valeurs[i].y,2,0,2*Math.PI,true)
            ctx.closePath();
            ctx.stroke();
        }
    }

    this.estimation=function(){
        var isIn=0
        for ( i = 0; i < this.valeurs.length; i++ ) {
            if(this.estDans(this.valeurs[i].x,this.valeurs[i].y))
                isIn++
        }
        return (Math.floor((4*isIn/this.valeurs.length)*100)/100)
    }
}
```

c.
```
<body onload="
    var n=prompt('Taille de la cible');
    var nb=prompt('Nombre de flèches');
    setInterval('init();lancer('+nb+','+n+')',1000)">
```

d.
```
function moyenne(){
   var somme=0
   for ( i = 0; i < listeEstimations.length; i++ ) {
      somme+= listeEstimations[i]
   }
   return ( Math.floor((somme/listeEstimations.length)*1000)/1000)
}
```

e.
```
function lancer(nombre,taille){
   var s=new Simulation(nombre,taille)

   s.dessinerCible()
   s.fleches()
   s.dessinerFleches()
   listeEstimations.push(s.estimation())
   document.getElementById("estim").innerHTML="Estimation:"+s.estimation()+""
   document.getElementById("moyenne").innerHTML="Moyenne:"+moyenne()+""
}
```

f.
```
div{
   font-size:3em;
   font-family:Arial;
   color:brown;
}
```