

Jeudi 12 décembre 2013

Directives:

1. Vous avez le droit à 2 pages (US letter) de notes. Les calculatrices électroniques, ordinateurs, tablettes et autres appareils du genre sont **interdits**. Vous pouvez vous servir de votre téléphone cellulaire pour avoir l'heure.
2. Le signe % en début de ligne dans les exemples de code indique un résultat d'exécution.
3. Inscrivez tout de suite votre **nom**, votre **code permanent** et le **numéro de votre place** (s'il existe).
4. Répondez **sur le questionnaire**, dans l'espace libre qui suit chaque question. Si vous manquez de place, écrivez au verso en l'indiquant très clairement. N'hésitez pas à commenter vos codes (en cas d'erreurs, cela peut vous sauver des points).
5. Le barème est donné à titre indicatif seulement. Votre note dépendra entre autre de l'élégance et de la simplicité de vos solutions.
6. **Conseil:** ne restez pas bloqué sur une question. La question 5 est la plus compliquée.

BONNE CHANCE!!

1.	_____	/15
2.	_____	/15
3.	_____	/15
4.	_____	/15
5.	_____	/15
6.	_____	/20
Total	_____	/ 95

Nom: _____ Code permanent: _____

Numéro de votre place: _____

1. (15 points) — Échauffement

Dans cette question, si vous pensez qu'un code est incorrect et ne peut s'exécuter, indiquez le clairement et précisez la nature du problème.

(a) (2 points) Que produit l'exécution de: `print (2+3+"="+2+3) ;`

(b) (2 points) Que produit l'exécution de: `print (2*3+"="+2*+3) ;`

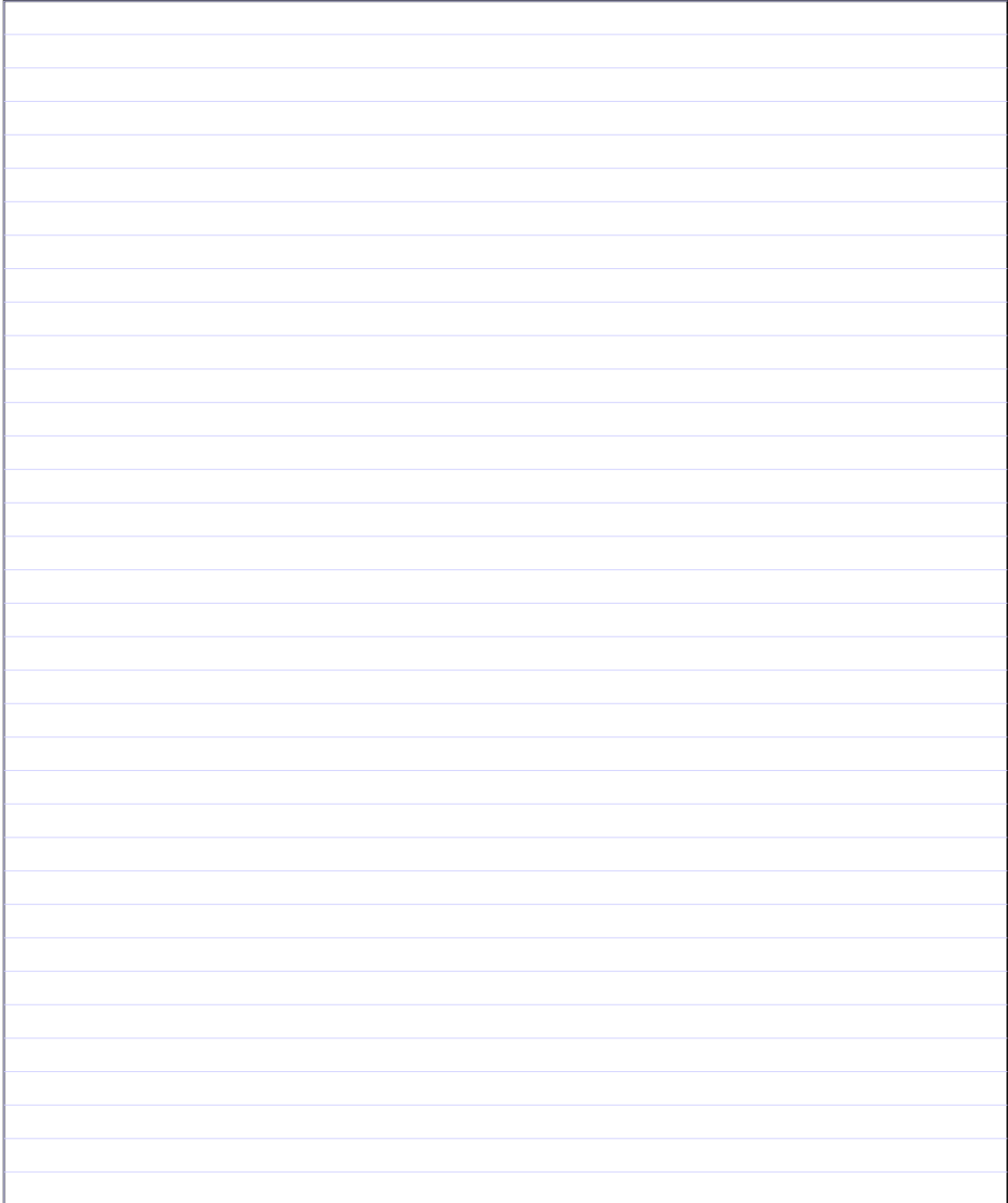
(c) (2 points) Que produit l'exécution de: `print (3-2+1-2*3) ;`

(d) (2 points) Que produit l'exécution de: `print ((Math.random() < 1)? true:false) ;`

(e) (2 points) Que produit l'exécution de: `print (["a", "b", 3]) ;`

2. (15 points) — Boucles simples

- (a) (5 points) Écrire une fonction `nbPair` qui prend en argument un tableau d'entiers (ou plus exactement son adresse) et qui retourne le nombre d'entiers pairs¹ dans ce tableau. L'appel suivant doit par exemple retourner 3: `nbPair([1, 2, 4, 9, 6])`. Il vous appartient d'écrire du code pour décider si un nombre (entier) est pair.



¹Un entier pair est divisible par 2!

3. (15 points) — Simplifiez-vous la vie

- (a) (3 points) Simplifiez le plus possible la fonction `quiz2` sans en modifier la sémantique (c'est-à-dire que votre fonction doit retourner le même résultat que `quiz2` pour les mêmes arguments). En particulier, vous ne devriez avoir qu'un seul `return`. Vous n'avez bien sûr pas le droit d'utiliser `break`, `continue` ou `goto` qui ne simplifieraient de toute façon pas le code. Il n'est pas non plus nécessaire d'introduire une nouvelle variable (le faire serait une mauvaise idée).

```
var quiz2 = function(t, j) {
  for (var i=0; i<t.length; i++)
    if (i === j)
      return t[j];
  return -1;
}
```

- (b) (3 points) Les deux fonctions suivantes sont-elles équivalentes. C'est-à-dire, produisent-elles le même résultat sur les mêmes entrées? Justifiez.

<pre>var quiz3a = function(a,b) { if ((a === 4) && (b === 5)) return a+b; else { if (b === 5) return a+b; return (a===4)? a:b; } }</pre>	<pre>var quiz3b = function(a,b) { if (a===4) { if (b===5) return a+b; else return a; } return a; }</pre>
--	--

(c) (4 points) Les deux fonctions suivantes sont elles équivalentes? Justifiez.

```
var quiz4a = function(a,b) {
  if (a === b) {
    if (a === 4)
      return true;
  }
  else
    if (a < b)
      return true;
}

var quiz4b = function(a,b) {
  if (a < b)
    return true;
  return (a === 4) && (b === 4);
}
```


(d) (5 points) Ces deux fonctions retournent-elles le même résultat quelque soit la valeur de i strictement positive passée en argument? Justifiez.

```
var quiz5a = function (i) {
  var sum = 0;
  do {
    if ((i % 2) === 0)
      i += 2;
    else
      i += 1;
    if (i > 10) i = 0;
    else
      sum += i;
  } while (i >= 1);
  return sum;
};

var quiz5b = function (i) {
  var sum = 0;
  if ( (i % 2) === 1 ) {
    ++i;
    sum += i;
  }
  for ( ; i <= 10; i += 2)
    sum += i;
  return sum;
};
```


4. (15 points) — Récursivité

Considérez le code suivant où `helper` est une fonction interne à `recurs`:

```
var recurs = function(s) {  
  
    var helper = function(s,i,j,n) {  
  
        if (i <= j) {  
            var ci = s.charAt(i);  
            var cj = s.charAt(j);  
  
            if (n) {  
                return ci + helper(s,i-1,j+1,!n) + cj;  
            }  
            return cj + helper(s,i+2,j-2,!n) + ci;  
        }  
        return "x";  
    }  
  
    return helper(s,0,s.length-1,false);  
  
};
```

(a) (2 points) La fonction `helper` est-elle récursive terminale ? Justifiez.

<hr/> <hr/> <hr/>

(b) (1 point) Que retourne l'exécution de: `recurs("a")` ?

<hr/> <hr/> <hr/>

(c) (2 points) Que retourne l'exécution de: `recurs("abcd")` ?

<hr/> <hr/> <hr/>

(d) (3 points) Que retourne l'exécution de: `recurs("abcdefg")` ?

<hr/> <hr/> <hr/> <hr/> <hr/>

- (e) (3 points) Écrire une fonction récursive qui implémente la récurrence qui suit pour tout entier n et m positifs (ou nuls):

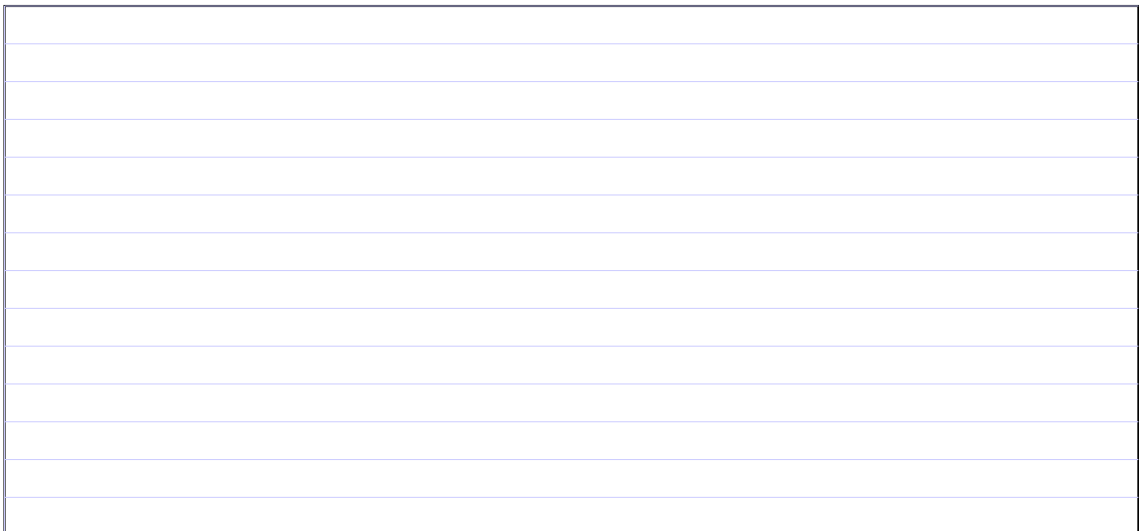
$$S(m, n) = \begin{cases} n & \text{si } m = 0 \\ 1 + S(m - 1, n) & \text{sinon} \end{cases}$$



- (f) (2 points) Que calcule la récurrence décrite dans la question précédente?



- (g) (3 points) Écrire une fonction récursive `entier(d, f)` qui affiche les entiers entre d et f inclus, sans utiliser de boucle.



(b) (8 points) Écrire une fonction losange (n) qui prend un entier n (strictement positif) en argument et qui affiche un losange de 2n+1 lignes, dont les appels suivants illustrent l'aspect:

```
losange (2) ;
% *
% ***
%*****
% ***
% *
```

```
losange (3) ;
% *
% ***
% *****
%*****
% *****
% ***
% *
```

```
losange (4) ;
% *
% ***
% *****
% *****
%*****
%*****
% *****
% *****
% ***
% *
```

6. (20 points) — POO

Vous allez dans ce problème doter javascript d'un objet `Matrice` qui permet de représenter des matrices et offre des opérations de base sur celles-ci. Lisez tout le problème avant de répondre aux différentes questions. Vous pouvez ajouter autant de méthodes que vous le voulez à votre objet `Matrice` de manière à rendre votre code le plus lisible possible.

(a) (2 points) Décrire brièvement les variables d'état d'un objet `Matrice`.

(b) (4 points) Un utilisateur souhaite créer des objets à l'aide des commandes suivantes:

- `new Matrice(n, m)` qui crée une matrice de dimension $n \times m$ qui contient des valeurs (entières) aléatoires choisies entre 1 (inclus) et 10 (inclus).
- `new Matrice(n, m, d, f)` qui crée une matrice de dimension $n \times m$ qui contient des valeurs (entières) aléatoires choisies entre `d` (inclus) et `f` (inclus).

Écrire le constructeur des objets `Matrice`.

- (c) (3 points) Ajouter à l'objet `Matrice` la méthode implicitement spécifiée par ce code et son exécution:

```
var mat1 = new Matrice(3,4,0,5);
print(mat1);
% 4      1      0      2
% 3      3      5      0
% 2      5      4      1
```


- (d) (3 points) Ajouter à l'objet `Matrice` la méthode `column(i)` qui retourne la *i*-ème colonne de la matrice concernée sous forme d'un tableau à une dimension. La première colonne est à l'indice 0, la seconde à l'indice 1, etc. Par exemple l'appel `mat1.column(1)`, où `mat1` est la matrice décrite à la question précédente, doit retourner le tableau `[1, 3, 5]`.

