Université Henri Poincaré — **Année 2008/2009**
Faculté des Sciences et Techniques — **Licence MI 1ère année**
CESS Épinal — **Semestre 1 - 1ère session**

TP noté – Algorithmique et Programmation 1
(Responsable : Yannick Parmentier)
4 Décembre 2008

Durée : 1 heure 45
Documents et calculatrices autorisés.
Barême donné à titre indicatif.

**** Veillez à systématiquement donner le type de vos fonctions et à les tester. ****

---

1 Approximation de e^x (8 points)

Sachant que l'exponentielle d'un nombre x peut être approchée par la série suivante :

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \dots$$

1. Proposez et implantez une fonction `serie_exp` permettant de calculer la série ci-dessus à un certain rang n.
   De quels paramètres dépend cette fonction, quel est son type ?

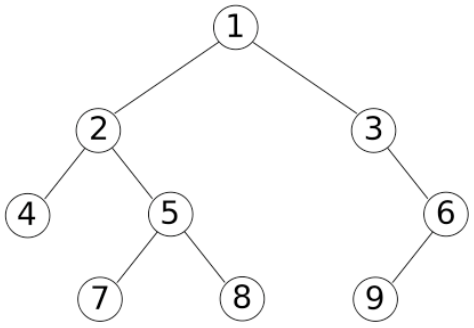
2. Définissez une fonction `diff` calculant la différence entre la fonction `Caml` prédéfinie `exp` et la fonction `serie_exp` que vous avez définie pour un certain réel `x` et un certain rang `n`.

   Complétez le tableau ci-dessous :

n :	5	10	20	30
$x = 0.$				
$x = 1.$				
$x = 2.$				
$x = 5.$				
$x = 10.$				

2 Modélisation des arbres binaires (12 points)

Dans cet exercice, on s'intéresse aux arbres binaires. Un arbre binaire est une structure de donnée composée de nœuds liés entre eux. Plus précisément, chaque nœud a au plus un père, et un seul nœud n'a pas de père, ce nœud est appelé racine de l'arbre. En outre, chaque nœud a 0, 1 ou 2 fils. Un nœud n'ayant pas de fils est appelé nœud feuille. La figure ci-dessous illustre cela.



Une telle structure peut être définie par le type récursif ci-dessous :

```
type arbre_binaire =  Vide
                    | Feuille of int
                    | Noeud of int * arbre_binaire * arbre_binaire;;
```

En d'autre termes, un arbre binaire est soit composé d'un nœud contenant un élément (ici nous prendrons comme élément un nombre entier), soit composé d'un sous-arbre gauche et d'un sous-arbre droit.

1. Définissez la variable `arbre1` associée à l'arbre binaire représenté sur la figure ci-dessus.

2. Écrivez une fonction `parcours_infixe`, qui, à partir d'un arbre binaire, retourne la liste des noeuds le composant, ordonnée de telle manière que le fils gauche d'un nœud apparaisse avant ce nœud, qui lui-même apparaît avant son fils droit.

   Exemple : le parcours infixe de l'arbre ci-dessous retournera la liste de nœuds
   `[4 ; 2 ; 7 ; 5 ; 8 ; 1 ; 3 ; 9 ; 6]`

3. Écrivez une fonction `appartient` qui vérifie si un nombre entier donné en paramètre est présent dans un nœud d'un arbre binaire également donné en paramètre.

4. Écrivez une fonction `poids` qui, à partir d'un arbre binaire donné en paramètre, retourne la somme des nombres des nœuds qui le composent.

5. Écrivez une fonction `equilibre` qui, à partir d'un arbre binaire donné en paramètre, retourne 0 si le nombre de son fils gauche est égal à celui de son fils droit, -1 si le fils gauche est supérieur, 1 si le fils droit est supérieur.

6. Nous appelons *arbre binaire équilibré* un arbre binaire dans lequel chaque nœud est équilibré. En d'autre termes, tout nœud est tel que son nombre est supérieur à celui de son fils gauche et inférieur à celui de son fils droit.

   Écrivez une fonction `abe` qui retourne vrai si l'arbre binaire donné en paramètre est un arbre binaire équilibré, et faux sinon (NB : l'arbre donné sur la figure ci-dessus est équilibré).

7. Écrivez une fonction `ajout` qui teste si un arbre binaire passé en paramètre est un arbre binaire équilibré, et si oui, ajoute un entier passé en paramètre à la bonne place dans l'arbre. Si l'arbre passé en paramètre n'est pas un arbre binaire équilibré, cette fonction retourne un message d'erreur.