# Colle Info. 10 - Révisions

## Exercice 09.1

Un jeu est une succession de jets d'une pièce qui fait pile avec la probabilité $p$. Un joueur dispose initialement d'une fortune $a$. On note $F_n$ la variable aléatoire égale à la fortune du joueur à l'issue du $n$-ième lancer. On convient que $F_0$ est la variable aléatoire certaine égale à $a$. On obtient la fortune $F_{n+1}$ à partir de $F_n$ de la manière suivante :
Avant le lancer $n+1$, le joueur mise une partie $X_n$, entière de sa fortune sur pile et l'autre partie, $F_n - X_n$ sur Face. Si le lancer $n+1$ fait apparaître pile, la fortune $F_{n+1}$ est égale à $2X_n$, s'il fait apparaître face, la fortune $F_{n+1}$ est égale à $2(F_n - X_n)$. Ainsi, à tout instant, la fortune du joueur est un entier pair, éventuellement nul.

On suppose que la loi conditionnelle de la variable $X_n$ sachant $[F_n = 2k]$ est une loi binomiale de paramètres $2k$ et $r$, étant un réel de $]0,1[$.
On considère le programme suivant :

```
PROGRAM simulation ;
VAR a,n,i,X,F : INTEGER ;
    r,p : REAL ;
FUNCTION mise(m : INTEGER ; s : REAL) : INTEGER ;
........................
END ;
BEGIN
RANDOMIZE ;
READLN(n) ; READLN(p) ; READLN(r) ; READLN(a) ; F:= a ;
FOR i := 1 TO n DO
    BEGIN
    X := mise(F,r) ;
    IF random < p THEN ..........
    .............................
    END ;
END.
```

La fonction `random` est une fonction sans argument. A son appel, l'ordinateur génère un nombre aléatoire compris entre 0 et 1, nombre qui suit une loi uniforme sur $[0,1]$. L'instruction `randomize` est utilisée pour obliger l'ordinateur à générer un nouveau nombre à chaque appel de la fonction.
La fonction `mise` est une fonction qui simule une loi binomiale de paramètres `m` et `s`. Elle doit donc prendre, à chaque appel, une valeur aléatoire entière comprise au sens large entre 0 et $m$, la probabilité qu'elle prenne une valeur donnée étant celle fournie par la loi binomiale de paramètres $m$ et $s$.

1. Rédiger les lignes manquantes (déclarations et instructions) dans la définition de la fonction `mise`
2. Rédiger les instructions manquantes du corps principal du programme de telle sorte que celui-ci calcule et affiche les fortunes succesives $F_0, \dots, F_n$ du joueur, les paramètres $a$, $r$, $p$, $n$ étant fournis par l'utilisateur.

**Exercice 09.2**

**Ecricome 2009 S**

Soient $a$ et $b$ deux entiers naturels non nuls et $N = a + b$. La suite $(u_n)$ donnée par l'énoncé vérifie

$$u_n = \left(1 - \frac{1}{100}\right) u_{n-1} + \frac{1}{10}$$

Ecrire une fonction en Pascal fournissant le calcul de $u_{2009}$.

**Exercice 09.3**

**Ecricome 2010 S**.

On introduit la suite $(u_n)$, définie par

$$\forall n \geqslant 1,\ u_n = \left(\sum_{i=1}^{n} \frac{1}{i}\right) - \ln(n)$$

'Ecrire un programme qui calcule $u_n$ pour un entier $n \geqslant 1$ donné.

**Exercice 09.4**

**Ecricome 2011 S**

On considère un tableau $X$ de nombres réels de taille 2011 préalablement rempli

```
X : ARRAY[1..2011] OF REAL
```

1. Ecrire une fonction en Pascal calculant et affichant les réels

$$\max(X[1], X[2]) \text{ et } \max(X[1], X[2], X[3])$$

2. Ecrire une fonction en Pascal calculant et affichant le réel

$$\max(X[1], X[2], \ldots, X[2011]) = \max_{1 \leqslant i \leqslant 2011}(X[i])$$

**Exercice 09.5**

**Ecricome 2010 E**

L'énoncé étudie la fonction $\varphi$ définie sur $\mathbb{R}^{+*}$ par :

$$\varphi(x) = \ln(x) - \ln(x+1) + \frac{1}{x}$$

On a préalablement montré que l'équation $\varphi(x) = 1$ possède une unique solution $\alpha$ et que $\dfrac{1}{3} < \alpha < \dfrac{1}{2}$.
Proposer un programme en Pascal permettant d'encadrer $\alpha$ dans un intervalle d'amplitude $10^{-2}$.

**Exercice 09.6**

**EM Lyon 2010 E**

Soit $f : x \mapsto x - \ln(1 + x^2)$ et $(u_n)$ la suite définie par

$$u_0 = 1,\ \forall n \in \mathbb{N}, u_{n+1} = f(u_n)$$

On a montré préalablement montré que la suite $(u_n)$ est décroissante et tend vers 0.
Ecrire un programme qui calcule et affiche un entier $n$ tel que $u_n \leqslant 10^{-3}$.

**Exercice 09.7**

**HEC 2009 E**

Soit $(u_n)$ la suite définie par $u_0 = 0,\ u_1 = 1,\ \forall n \in \mathbb{N}, u_{n+2} = u_{n+1} + u_n$. On propose la fonction Pascal suivante :

```
FUNCTION f(n : INTEGER) : INTEGER ;
VAR temp,u,v,k : integer ;
BEGIN
    u := 0 ; v := 1 ;
    FOR k := 1 TO n-1 DO
        BEGIN
        temp := _____ ; v := _____ ; u := _____ ;
        END ;
    f := _____ ,
END ;
```

Compléter cette fonction aux quatre places signalées par des tirets de façon que la valeur rendue soit $u_n$.

**Exercice 09.8**

**HEC 2009 S**

On considère la suite $(s_n)$ définie par

$$s_1 = 1,\ s_2 = \frac{4}{5},\ s_3 = \frac{2}{5},\ \forall n \in \mathbb{N}^*, s_{n+3} = \frac{3}{2}s_{n+2} - s_{n+1} + \frac{1}{4}s_n$$

Ecrire une fonction Pascal d'en-tête `suite(s1,s2,s3 : REAL, n : INTEGER) : REAL` qui, pour tout $n$ de $\mathbb{N}^*$, renvoie le $n$-ième terme de cette suite.

**Exercice 09.9**

**Ecricome 2011 E**

On se donne $\varphi(x) = 1 - x^2 \ln(x)$ si $x \neq 0$ et $\varphi(0) = 1$. On considère les deux suites $(a_n)$ et $(b_n)$ définies par

$$a_0 = \sqrt{2},\ b_0 = 2$$

$$\forall n \geqslant 0,\ \text{si}\varphi(a_n)\varphi\left(\frac{a_n + b_n}{2}\right) < 0,\ a_{n+1} = a_n,\ b_{n+1} = \frac{a_n + b_n}{2}$$

$$\forall n \geqslant 0,\ \text{si}\varphi(a_n)\varphi\left(\frac{a_n + b_n}{2}\right) \geqslant 0,\ a_{n+1} = \frac{a_n + b_n}{2},\ b_{n+1} = b_n$$

Ecrire un programme en Pascal calculant $a_n$ et $b_n$ pour un entier $n$ donné par l'utilisateur.

**Exercice 09.10**

**HEC 2002 S**

Soit $A = (a_{i,j}) \in \mathcal{M}_{n,p}(\mathbb{R})$ une matrice. On note $u(A) = \min\limits_{1\leqslant i\leqslant n}(\max\limits_{1\leqslant j\leqslant p} a_{i,j})$. On suppose que dans le préambule d'un programme écrit en Pascal, on a défini deux constantes entières : `n` et `p` et un type `matrice = ARRAY[1..n,1..p] OF REAL ;`

1. Ecrire une fonction `MaxLigne (A : matrice ; i : integer) : real ;` cette fonction doit retourner le plus grand élément de la ligne `i` de la matrice `A`.
2. Ecrire une fonction `MinMax(A : matrice) : real ;` cette fonction doit retourner la valeur $u(A)$ ; on pourra utiliser la fonction `MaxLigne`.