Corrigé PASCAL

1 Suites récurrentes

1.1 ESC 2006 (IAF)

On veut $0 \leqslant u_k - \alpha_2 \leqslant \varepsilon$.
On peut l'obtenir par $0 \leqslant u_k - \alpha_2 \leqslant \left(\frac{1}{e}\right)^k \leqslant \varepsilon$
On résout cette dernière condition :.
$\left(\frac{1}{e}\right)^k \leqslant \varepsilon \Longleftrightarrow k \geq -\ln(\varepsilon)$ ce qui est vérifié pour le premier entier plus grna d que $-\ln(\varepsilon)$ à savoir sa partie entière $+1$, le N du programme.
Il y a donc simplement à calculer de u_1 à u_N

```
for k := 1 to N do u:=exp(u)-2 ;
```

1.2 ECRICOME 2006 (IAF)

On considère la fonction f définie pour tout réel x par : $.f(x) = x + 1 + 2e^x$
Donc $f'(x) = 1 + 2e^x$.
On peut, soit résoudre $\dfrac{1}{e^{2^n-1}} \leqslant 10^{-p} \Longleftrightarrow n \geqslant N$ et calculer les termes de 1 à N
soit calculer 10^{-p} puis u_n et $\dfrac{1}{e^{2^n-1}}$ jusqu'à ce que ce dernier soit $\leqslant 10^{-p}$:

```
writelen('p?');readln(p);
dix:=1;for i:=1 to p do dix:=dix*10; eps:=1/dix; { calcul 10 ^p }
pn:=1;u:=-1; {pn contient 2^n }
repeat
     u:=u+(u+1+2*exp(u))/(1+2*exp(u));
     pn:=pn*2;
until exp(1-pn)<=eps; {évite de calculer un inverse }
writeln(u);
```

1.3 ECRICOME 2003 (élémentaire)

Pour calculer u_{10} on affecte à une même variable `u` les valeurs successives de u_0 à u_{10}.
Il faut donc calculer les valeurs suivantes de u_1 à u_{10} :
on a $f(u_n) = \dfrac{u_n}{sh(u_n)} = \dfrac{2u_n}{e^{u_n} - e^{-u_n}}$

```
program suite;
var u:real;n:integer;
begin
     u:=1;
     for n:=1 to 10 do u:=2*u/(exp(u)-exp(-u));
     writeln(u)
end.
```

1.4 EML 2002 (Dichotomie)

x_2 est solution de $P_2(x) = 0$.
Pour raccourcir les calculs que $P_2(x) = -x + x^2/2 - x^3/3 + x^4/4 = x\left[-1 + x\left(1/2 + x\left[-1/3 + x/4\right]\right)\right]$

On programme la méthode de dichotomie pour programmer le calcul de x_2 à 10^{-3} près.
x_2 est dans l'intervalle $[0,1]$.
Pour tout entier n,
Si x_2 est dans l'intervalle $[a_n, b_n]$, soit c_n le millieu : $c_n = \frac{a_n+b_n}{2n}$.
Si $P_2(c_n)$ est du même signe que $P_n(a_n) > 0$ alors ils sont tout deux du même côté de x_2 et celui ci se trouve donc dans $[c_n, b_n]$ d'où le nouvel encadrement avec $a_{n+1} = c_n$ et $b_{n+1} = b_n$ et sinon, oil est dans l'ntervalle $[a_,, c_n]$

```
program approche;n
var a,b,c:real;
function p(x:real):real;
begin
     p:=x*(-1+x(1/2+x(-1/3+x/4)));
end;
begin
     a:=0;b:=1;
     repeat
          c:=(a+b)/2
          if p(c)>0 then b:=c else a:=c;
     until b-a<=1E-3;
     writeln(b);
end.
```

1.5 ESCP 2000 (récurrence à 2 termes)

On a besoin des deux termes précédents pour calculer le suivant.
`vs` pour le suivant `vp` le précédent et `vpp` le précédent du précédent.
Attention à l'ordre des affectation ! `vs:=sqrt(vp)+sqrt(vpp);vpp:=vp;vp:=vs;`
l'ancien suivant est le nouveau précédent. l'ancien précédent est le nouveau précédent du précédent.
Si on fait `vs:=sqrt(vp)+sqrt(vpp);vp:=vs;vpp:=vp;` lors de cette affectation, `vp` ne contient plus le précédent !

k	v_{k+2}	v_{k+1}	v_k
n	v_{n+2}	v_{n+1}	v_n
$n+1$		v_{n+2}	v_{n+1}

On a à calculer les termes de v_2 à v_N

```
Program Suite;
var vp,vpp,vs:real;i,N:integer;
begin
writeln('v0 et v1 ?');readln(vpp,vp);
writeln('N?');readln(N);
for i:=2 to N do
begin
     vs:=sqrt(vp)+sqrt(vpp);
     vpp:=vp;vp:=vs;
     writeln(vs);
end;
end.
```

1.6 EDHEC 1999 (récurrence à 2 termes)

$a_1 = 1$ et $b_1 = 1$

(...) On montrera que : $\forall n \in \mathbb{N}^*, a_{n+1} = \dfrac{a_n}{n+1} - \dfrac{b_n}{(n+1)^2}$ et $b_{n+1} = \dfrac{b_n}{n+1}$

On a besoin des valeurs précédentes de a_n et de b_n pour calculer les suivantes.
`ap` et `bp` pour a_n et b_n; `as` et `bs` pour a_{n+1} et b_{n+1}.
On a aussi besoin de l'indice précédent ('n').
Donc on calcul pour les indices précédents de 1à $N-1$

```
Program Suite;
var ap,bp,as,bs:real;i,N:integer;
begin
writeln('N?');readln(N);
ap:=1;bp:=1;
writeln(ap, bp);
for i:=1 to N-1 do
begin
     as:=ap/(i+1)-bp/sqrt(i+1);
     bs:=bp/(i+1);
     ap:=as;bp:=bs;
     writeln(ap,bp);
end;
end.
```

2 Sommes (Accumulateurs)

2.1 EDHEC 2006 (récursive)

1. On recopie la définitino récurrcive de la factorielle : $0! = 1$ et pour tout $n \geq 1 : n! = n \cdot (n-1)!$

```
Function f(n :  integer) :  integer;
Begin
If (n=0) then f:=1;
else f:  =f(n-1)*n;
end;
```

Compléter cette déclaration pour qu'elle renvoie $n!$ lorsqu'on appelle $f(n)$.

2. On considère la déclaration de fonction récursive suivante :

```
Function g (a:real ; n:integer):real;
Begin
If (n=0) then g:  = 1
else g:  = a * g(a,n- 1);
end ;
```

On a pour tout $n \geq 1 : g(a,n) = a \cdot g(a, n-1)$ suite géométrique donc $g(a,n) = 1 \cdot a^n$

3. Proposer un programme (sans écrire la partie déclarative) utilisant ces deux fonctions et permettant d'une part le calcul de la somme $\sum_{k=0}^{n-1} \dfrac{a^k}{k!} e^{-a}$

```
begin
writelen('n?');readln(n);
S:=0;
for i:=0 to n-1 do S:=S+g(a,i)/f(i);
S:=S*exp(-a); {on ne fait qu'une seule fois le produit, à la fin}
writeln(S);
end.
```

4. Compléter la déclaration de fonction suivante pour, qu'elle renvoie la valeur de $\sum_{k=0}^{n-1} \frac{a^k}{k!} e^{-a}$ à l'appel de *sigma(a,n)*.

```
Function sigma(a :  real ; n :  integer) :  real;
var k :  integer;
p:  real;
Begin
p :  = 1 ; s:=1;
For k:  = 1 to n-1 do begin p := p*a / k; s:=s+p end;
s:= s*exp(-a);
sigma:  = s;
end;
```

N.B. cette méthode est beaucoup plus économe en calculs. La première reprend à 0 le calcul de la puissance et de la factorielle à chaque terme de la somme.
La seconde utilise le fait que $\frac{a^k}{k!} = \frac{a^{k-1}}{(k-1)!} \cdot \frac{a}{k}$ qui ne demande qu'un produit et un quotient pour déterminer le terme suivant à ajouter.

3 PASCAL Probabilités

Complément

On simule les événements par des événements `r:=random; (a<r and r<b)` de même propriétés : probabilité `a-b,` incompatibilité ...
Randomize ne doit être utilisé qu'une seule fois, en début de programme.

3.1 D'après EML 2004

Une urne contient des boules blanches, des boules rouges et des boules vertes.

- La proportion de boules blanches est b.
- La proportion de boules rouges est r.
- La proportion de boules vertes est v.

Ainsi, on a.: $0 < b < l, \quad 0 < r < l, \quad 0 < v < 1 \quad$ avec $\quad b + r + v = 1$.
On effectue des tirages successifs avec remise et on s'arrête au premier changement de couleur.

1. Ecrire une fonction qui donne 1 avec une probailité de b, 2 avec une probabilité de r et 3 avec une probablité de $1 - b - r$.

Pour `x:=random`, les événements `(x<b)` et `(b<=x and x<(b+r))` sont incompatibles et de probabiltiés `b` et `r`

On pourrait aussi prendre `(x<b)` et `(1-r<x)`

```
function aleat(b,r:real):integer;
var :real;
begin
x:=random;
if (x<b) then aleat:=1;
if (b<=x and x<(b+r)) then aleat:=2;
if (b+r)<x then aleat:=3;
end;
```

2. D'après le `writelen` c'est `k` qui compte le nombre de tirages. D'où le `k:=1` au premier puis le `k:=k+1` dans la boucle.

Complèter le programme suivant pour qu'il affiche le rang du premier changement de couleur :

`x:=aleat(r,b);` simule le premier lancer.

On aura le premier changement quand un tirage sera différent de ce premier lancer (plus simple que de tester par rapport au précédent qui obligerait à mettre à jour le précédent tirage)

D'où le test sur `x=y` avec `y` qui contient le résultat du lancer courant.

```
begin
writelen('proportion de rouges et de blanches ?');
readln(r,b);
randomize;
x:=aleat(r,b);
```

```
k:=1;
repeat k:=k+1 ; y:=aleat(r,b) until x=y;
writeln( 'il a fallut ',k,'lancers');
end.
```

3.2 EDHEC 2004

On lance n fois une pièce équilibrée (cest-à-dire donnant pile avec la probabilité 1/2 et face également avec la probabilité 1/2), les lancers étant supposés indépendants.
On note Z la variable aléatoire qui vaut 0 si l'on n'obtient aucun pile pendant ces n lancers et qui, dans le cas contraire, prend pour valeur le rang du premier pile. (pile sera codé par le nombre 1 et face par 0).

On reconnait à `k:=k+1` que `k` est le compteur du nombre de lancers.
Dans le programme, si "pile" soit `(lancer=1)` n'est jamais réalisé, `z` conserve la même valeur : 0 sa valeur de départ.
Si on a pile, `z:=k` prend comme valeur le nombre de lancers effectués.
La boucle s'arrête quand on a obtenu pile ou quand on a effectué les n lancers.

```
Program EDHEC2004 ;
var k, n, z, lancer :  integer ;
Begin
     Randomize ;
     Readln(n) ; k := 0 ; z := 0 ;
     Repeat
          k := k + 1 ; lancer := random(2) ;
          If (lancer = 1) then z:=k ;
     until (lancer = 1) or (k=n) ;
     Writeln (z) ;
end.
```

3.3 HEC II 2004

Deux joueurs J et J' s'affrontent dans un jeu utilisant la même expérience aléatoire que précédemment avec les règles suivantes :

- le joueur J est gagnant si la configuration " pile, pile, face " apparaît dans la suite des résultats des lancers, avant que la configuration " face , pile, pile " n'apparaisse;
- le joueur J' est gagnant si la configuration " face, pile, pile " apparaît dans la suite des résultats des lancers, avant que la configuration " pile, pile, face " n'apparaisse;

1. Donner sous forme d'un tableau les valeurs successives prises par les variables x, y et k lors de l'exécution de cette procédure, si les valeurs données à la variable r par la fonction " RANDOM(2)" sont successivement :

a)

random		1	1	1	1	0
x	0	1	2	2	2	3
y	0					1
k	0	1	2	3	4	5

b)

random		1	0	1	0	0	0	1	1
x	0	1	0	1	0	0	0	1	2
y	0		1	2	1	1	1	2	3
k	0	1	2	3	4	5	6	7	8

c)

random		0	1	0	1	0	1	1
x	0	0	1	0	1	0	1	2
y	0	1	2	1	2	1	2	3
k	0	1	2	3	4	5	6	7

2. La valeur de k est celle du nombre de lancers nécessaires pour obtenir le premier PPF ou FPP

En effet, en associant P à la valeur random=1 et F à la valeur random=0, x compte combien on a de termes de la séquence PPF et y combien de termes de FPP :

- Les lancers s'arrètenet quand on a une triplette. (PPF ou FFP $x = 3$ ou $y = 3$)
- Quand on a Pile,
  - pour x :
    si on au moins un terme, (et pas 3) c'est que l'on a PP et on conserve ce début
    si on a aucun terme, on a P et donc le premier terme de la séquence.
  - pour y :
    si on a un terme c'était F donc on a à présent 2 termes et si on avait 2 termes, c'étaient FP et on en a trois.
    dans les deux cas, y augment de 1.
- Quand on a Face,
  - pour x :
    si $x = 2$, on avait alors PP et on clos la séquance avec F et $x = 3$
    sinon, on avait un seul pile et on n'a plus aucon terme de la séquence PPF donc $x = 0$
  - pour y :
    F marque le début d'une séquence FPP donc $y = 1$

Donc on complète par :

```
IF x=3 THEN WRITE ('PPF avant FFP ') ELSE WRITE('FPP avant PPF');
```

Et l'ordinateur afficherait PPF avant FPP dans le premier exemple et l'inverse dans les deux autres.

3.4 ESSEC 2003

On effectue des lancers successifs (indépendants) d'un dé cubique équilibré, dont les faces sont numérotées de 1 à 6, et on note $X_1, X_2, \ldots X_n, \ldots$, les variables aléatoires donnant le numéro amené par le dé aux premier lancer, deuxième lancer, ...

Pour tout entier naturel n non nul, on note Y_n, la somme des points obtenus aux n premiers lancers.

Enfin, pour tout entier naturel k non nul, la variable aléatoire T_k compte le nombre de celles des variables aléatoires $Y_1, Y_2, \ldots, Y_n, \ldots$ qui prennent une valeur inférieure ou égale à k .

Par exemple, si les cinq premiers numéros amenés par le dé sont, dans l'ordre : 3, 1, 2, 3, 6, alors les événements suivants sont réalisés : $(Y_1 = 3)$, $(Y_2 = 4)$, $(Y_3 = 6)$, $(Y_4 = 9)$, $(Y_5 = 15)$, et les variables aléatoires $T_{\ 2}$, $T_{\ 3}$, $T_{\ 9}$ et $T_{\ 12}$ prennent respectivement pour valeurs 0, 1, 4 et 4 .

1. Simulation informatique

Compléter les lignes marquées par les symboles . . . du programme Pascal ci-dessous, de façon qu'il simule l'expérience aléatoire étudiée et affiche la valeur de T_{12}.

`x` contient le résultat du lancer du dé.

`y` acumule les lancers d'où le `y:=0` du début et le `y:=y+x;` à chaque boucle.

`t` compte le nombre de lancers jusqu'à dépasser 12 : `until t>12;`

Le nombre de lancers avant de dépasser 12 est alors un de moins d'où le `writeln(T=',t-1);`

```
Program ESSEC2003A;
var x,y,t:integer;
begin
     randomize;
     y:=0;t:=0;
     repeat
          x:=random(6)+1;
          y:=y+x;
          t:=t+1;
     until t>12;
     writeln(T=',t-1);
end.
```

3.5 ESSEC 2003

1. On se propose de simuler informatiquement une variable aléatoire.

On supposera que random(3) fournit au hasard un nombre élément de $\{1,2,3\}$ et que random(2) fournit au hasard un élément de $\{1,2\}$.

N.B. ce n'est pas la norme de de PASCAL !!!

```
program ESSEC2003
var ini,y :  integer
begin
ini:=random(3);
if ini=3 then y:=random(2) else y:=3 ;
end.
```

Si `ini=3` $Y=1$ et $Y=2$ sont équiprobables. Donc $\mathrm{P}_{ini=3}(Y=1)=\mathrm{P}_{ini=3}(Y=2)=\frac{1}{2}$ et $\mathrm{P}_{ini=3}(Y=3)=0$

Sinon, $Y=3$, donc $\mathrm{P}_{ini\neq 3}(Y=3)=1$ et $\mathrm{P}_{ini\neq 3}(Y=1)=\mathrm{P}_{ini\neq 3}(Y=2)=0$

Donc (par les proabilités totales ou par décomposition de l'événement) la loi de Y est :

$Y(\Omega)=\{1,2,3\}$ et $\mathrm{P}(Y=3)=\frac{2}{3}$ et $\mathrm{P}(Y=1)=\mathrm{P}(Y=2)=\frac{1}{6}$

et $E(Y)=\frac{1}{6}+\frac{2}{6}+\frac{3\cdot 2}{3}=\frac{5}{2}$