1 EDI de Turbo PASCAL

(Environnement de développement intégré)

1. Les touches spéciales du clavier :

   - **Alt+Tab** permet de basculer d'une application à une autre, **Alt+Entrée** permet de basculer l'éditeur Turbo Pascal de plein écran en une fenêtre windows.
   - verrNum (ou Num Lock) permet de commuter le pavé numérique en pavé de flèches.
   - CapsLock (le cadenas) permet de BLOQUER LE CLAVIER EN MAJUSCULE. (déblocage avec CapsLock)
   - Les touches Maj, Ctrl, Alt et AltGr s'utilisent simultanément avec une autre touche (d'abord Maj, puis, en la tenant enfoncée, la toutche 5([ )
   - Le symbole du haut est accessible avec Maj, celui du dessous en nomal, et celui de droite, quand il existe, pour ˜#{[|‘\ˆ@ } avec la touche ALtGr
   - La touche Alt sert de raccourci pour accéder aux item des menus : Alt+F pour accéder au menu File.
   - Supr détruit le caractère précédent, et ← détruit le caractère précédent.
   - Dans l'éditeur de TurboPascal, la touche de Tab de tabulation provoque l'alignement sur le paragraphe précédent

2. Pour créer un programme on utilise l'éditeur de texte de Turbo Pascal, on sauve le programme et enfin, on l'exécute et on peut en voir le résultat.

   dans debug/user Screen (alt F5)

3. Pour commencer un programme :

   lancer Turbo Pascal depuis le poste de travail

   Faire File/Change dir et vérifier que vous êtes bien dans **votre** répertoire.

   Sinon sélectionner le votre **cliquer sur Chdir** puis sur OK et faire file/new pour créer un document qui sera sauvé dans **votre répertoire.**

4. **Avant d'exécuter** votre programme, commencez toujours par le sauver (touche F2)

   Pour exécuter le programme taper sur Ctrl F9

5. pour voir le réultat d'un programme clique sur Debug/User Screen (**alt F5**)

   et pour avoir en permanence la fenêtre d'édition et la fenêtre des résultats, cliquer sur Debug/output

2 Suites et Sommes

2.1 Suivant, précédent.

Dans une même variable, on range successivement plusieurs valeurs. C'est à vous de savoir ce qui s'y trouve à chaque instant (le suivant ou le précédent ?). Vérifier particulièrement les valeurs stockées au début et à la fin des boucles.

Pour calculer $\sum_{k=1}^{n} \frac{1}{k!}$ on a $\sum_{k=1}^{n+1} \frac{1}{k!} = \sum_{k=1}^{n} \frac{1}{k!} + \frac{1}{(n+1)!}$ avec $(n+1) = (n+1)\,n!$
que l'on comprend ainsi :
La factorielle suivante est le produit de la précédente et de l'indice suivant.
la somme suivante est le total de la somme précédente et de la factorielle suivante.

- `F:=1; S:=1/F;`

  `for n:=1 to 10 do begin F:=F*n;S:=S+1/F end;`

  `writelen(S);`

  au début, `F` contient 0! et `S`= $\frac{1}{0!} = \sum_{k=0}^{0} \frac{1}{k!}$

  pour $n = 1$: `F:=F*n` calcule bien la factorielle suivante et `S:=S+1/F` la somme suivante.

  Pour $n = 10$ on aura donc ajouté les termes jusqu'à 1/10! et on aura $\sum_{k=0}^{10} 1/k!$ dans S à la fin.

- variante :

```
F:=1; S:=0;
for n:=1 to 10 do
begin
     S:=S+F; {F contient la !  précédente}
     F:=F/n
end;
writelen(S);
```

  Au début, `F` contient 0! et `S` = 0

  pour $n = 1$: `S:=S+F` calcule la somme pour 1/0!

  et `F:=F*n` calcule bien la factorielle suivante :1!....

  Pour $n = 10$ on fini par calculer 10! mais avant cela, on aura ajouté 1/9! à `S`

  Donc ce programme affecte $\sum_{k=0}^{9} 1/k!$ dans `S` à la fin.

2.2 Conditions d'arrêt, valeur approchée

$|u_n - \alpha| \leq \left(\frac{1}{2}\right)^n$ se traduit par u_n est une valeur approchée de α à $\left(\frac{1}{2}\right)^n$ près.
Pour calculer α à 10^{-3} près,il suffit de calculer u_n jusqu'à ce que $\left(\frac{1}{2}\right)^n \leq 10^{-3}$

```
Begin
     p:=1; u:=2;
     writeln('précision?'); readln(eps);
     repeat
          p:=p*0.5; u:=f(u);
     until p<=eps
```

```
    writeln(u);
End.
```

$u_n \leq \alpha \leq u_n + \varepsilon$ donc $0 \leq \alpha - u_n \leq \varepsilon$ se traduit par u_n est une valeur approchée de α à ε près.
(utilisé dans la méthode de dichotomie)

2.3 Compteur

Démarre à 0 et augmente de 1 chaque fois que nécessaire.
Cela s'utilise pour

- mémoriser l'indice d'une suite :

```
u:=u0;n:=0;
repeat n:=n+1;u:=f(u) until u>100
writeln(n);
```

- ou compter le nombre de "succès" d'une expérience :

```
randomize;
C:=0; writeln('probabilité de succès ?'); readln(p);
for i:=1 to 100 do if random<p then c:=c+1;
writeln('nombre de succès:  ',c);
```

2.4 Récursivité

Les fonctions et procédure de Pascal peuvent s'appeler elles mêmes,.
Cela permet de programmer directement la relation de récurrence mathématique :
N.B : l'ordinateur doit garder en mémoire (entasser) tous les résultats intermédiaire. On peut avoir ainsi des débordement de mémoire....

Exemple factorielle : $0! = 1$ et pour tout $n \geq 1$: $n! = n\,(n-1)!$
(factorielle suivante = indice suivant *factorielle précédente)

```
function fact(n:integer):integer;
begin
if n<0 then fact:=0;
if n=0 then fact:=1;
if n>0 then fact:=fact(n-1)*n;
end;
```

Exercice Coefficients du binôme :
$\binom{n}{p} = 0$ si $n < 0$ et si $p < 0$
$\binom{0}{0} = 1$ et $\binom{0}{p} = 0$ si $p \geq 0$
$\binom{n}{p} = \binom{n-1}{p-1} + \binom{n-1}{p}$ si $n > 0$
Ecrire une fonction récurrcive d'entête `function bin(n,p:integer):integer;`
qui calcule $\binom{n}{p}$

2.5 Récurrente $u_{n+1} = f\,(u_n)$

Algorithme :
`u:=u0 ;` et on répète `u:=f(u);` jusqu'à la condition d'arrêt ou pour les valeurs de n nécessaires.

2.6 Récurrences à deux termes

Il faut ici des variables pour les deux précédents et le suivant.

Exemple $u_0 = 1, u_1 = 2$ et $u_{n+2} = u_{n+1} + 2u_n$

Déterminer la première valeur de n pour laquelle $u_n \geq 10$:
On répète le calcul jusqu'à ce que $u_{n+2} \geq 10$ et l'indice est celui de u_{n+2}

```
u:=1;v:=2;n:=2;
repeat
n:=n+1;w:=v+2u; u:=v; v:=w;
until w>=10;
writeln(n);
```

Exercice Avec en plus l'indice dans la relation :

$u_1 = 1,\ u_2 = 2$ et $u_{n+2} = \frac{1}{n}u_{n+1} + 2u_n$ pour tout $n \geq 1$.
Calculer u_{10} (sera trouvé par $u_{10} = \frac{1}{8}u_9 + 2u_8$)

- ```
  u:=1; v:=2;
  for n:=1 to 8 do
  begin
      w:=v/n+2u; u:=v; v:=w;
  end;
  writeln(w);
  ```

2.7 EDHEC 2006 (récursive)

1. On considère la déclaration de fonction, en Pascal, rédigée de manière récursive :

```
Function f(n :  integer) :  integer;
Begin
If (n=0) then f:=...
else f:  =...
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 ;
```

Dire quel est le résultat retourné à l'appel de $g(a, n)$.
```

3. Proposer un programme (sans écrire la partie déclarative) utilisant ces deux fonctions et permettant d'une part le calcul de la somme $\sum_{k=0}^{n-1} \frac{a^k}{k!} e^{-a}$ et d'autre part, à l'aide du résultat de la question 1a), le calcul et l'affichage du taux de panne à l'instant n d'une variable aléatoire suivant la loi de Poisson de paramètre $a > 0$, lorsque n et a sont entrés au clavier par l'utilisateur (on supposera $n \geq 1$).

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 := ...; end;
s:= ...;
sigma:  = s;
end ;
```

2.8 ECRICOME 1999 second ordre

On considère la suite u définie par $u_0 = a$, $u_1 = b$ et pour tout $n \in \mathbb{N}$: $u_{n+2} = \sqrt{u_n} + \sqrt{u_{n+1}}$
Ecrire un programme en Turbo Pascal qui calcule et affiche la valeur de u_n pour des valeurs de a et b réelles supérieures ou égales à 1 et de n entier supérieur ou égal à 2, entrées par l'utilisateur.

2.9 ECRICOME 2007 (condition à résoudre)

Soit a un réel strictement positif. On considère la fonction f_a définie pour tout réel t strictement positif par :

$$f_a(t) = \frac{1}{2}(t + \frac{a^2}{t})$$

ainsi que la suite $(u_n)_{n\epsilon\mathbb{N}}$ de nombre réels déterminée par son premier terme $u_0 > 0$ et par la relation de récurrence :

$$\forall n \in \mathbb{N} \qquad u_{n+1} = f_a(u_n)$$

on montre que $|u_n - a| \leq \left(\frac{1}{2}\right)^{n-1} |u_1 - a|$

1. En déduire la convergence de la suite (u_n) et indiquer sa limite.

2. En utilisant ce qui précède, écrire un programme en langage Pascal permettant d'afficher les 100 premiers termes d'une suite (u_n), de premier terme 1, convergeant vers $\sqrt{2}$.

2.10 ESC 2006 (IAF Condition à résoudre)

Dans cette question on note $(u_k)_{k\in\mathbb{N}}$ la suite ainsi définie : $u_0 = -1$ et pour tout entier naturel k, $u_{k+1} = e^{u_k} - 2$ (....)
En déduire par récurrence sur k que pour tout entier naturel $k : 0 \leqslant u_k - \alpha_2 \leqslant \left(\frac{1}{e}\right)^k$.
On considère le programme Turbo-Pascal suivant: (où `trunc` désigne la fonction partie entière)

```
program ex2 ;
var N , k :  integer ; epsilon , u :  real ;
begin
     writeln ( ' Donnez un reel strictement positif');
     readln (epsilon );
     N := trunc ( - Ln (epsilon ) ) + 1 ; u := -1 ;
     for k := 1 to N do .........................  ;
     writeln(u);
end.
```

Montrer que l'entier naturel N calculé dans ce programme vérifie : $\left(\frac{1}{e}\right)^N \leq$ epsilon

Compléter la partie pointillée de ce programme afin que la variable `u` contienne après son exécution une valeur approchée de α_2 à epsilon près.

2.11 ECRICOME 2006 (IAF condition à résoudre)

On considère la fonction f définie pour tout réel x par : $.f(x) = x + 1 + 2e^x$
On s'intéresse à la suite $(u_n)_{n\in\mathbb{N}}$ définie par son premier terme $u_0 = -1$ et par la relation

$$\forall n \in \mathbb{N} u_{n+1} = u_n - \frac{f(u_n)}{f'(u_n)}$$

(...) on montre que $0 \leqslant u_n - \alpha \leqslant \dfrac{1}{e^{2^n-1}}$
Écrire un programme en langage Pascal permettant, lorsque l'entier naturel p est donné par l'utilisateur, de calculer une valeur approchée de α, de telle sorte que l'on ait :

$$0 \leqslant u_n - \alpha \leqslant 10^{-p}$$

2.12 ECRICOME 2003 (élémentaire)

On considère les fonctions ch et sh définies sur $\mathbb{R}$ par :

$$sh(x) = \frac{e^x - e^{-x}}{2} \text{ et } f(x) = \frac{x}{sh(x)}$$

On s'intéresse dans cet exercice à la convergence de la suite $(u_n)_{n\in\mathbb{N}}$ définie par la relation de récurrence :

$$\begin{cases} u_0 = 1 \\ \forall n \in \mathbb{N} \;\; u_{n+1} = f(u_n) \end{cases}$$

1. Ecrire un programme en Turbo-Pascal permettant de calculer et d'afficher u_{10}

2.13 EML 2002 (Dichotomie)

On considère, pour tout $n \in \mathbb{N}^*$, la fonction polynomiale $P_n : [0,+\infty[\longrightarrow \mathbb{R}$ définie pour tout $x \in [0,+\infty[$, par :

$$P_n(x) = \sum_{k=1}^{2n} \frac{(-1)^k x^k}{k} = -x + \frac{x^2}{2} + \ldots + \frac{-x^{2n-1}}{2n-1} + \frac{x^{2n}}{2n}$$

1. (...) on a montré que sur $[1,+\infty[$, P_n était strictement croissante et que $P_n(1) < 0$ et $P_n(2) \geq 0$
Montrer que, pour tout $n \in \mathbb{N}^*$, l'équation $P_n(x) = 0$, d'inconnue $x \in [1,+\infty[$, admet une solution et une seule notée x_n, et que :

$$1 < x_n \leqslant 2$$

2. Écrire un programme en langage Pascal qui calcule une valeur approchée décimale de x_2 à 10^{-3} près.

2.14 ESCP 2000 (récurrence à 2 termes)

On considère la suite $V = (v_n)_{n\geq 0}$ vérifiant $v_0 = 0$, $v_1 = \beta$ et, pour tout n positif, la relation

$$v_{n+2} = \sqrt{v_{n+1}} + \sqrt{v_n}$$

1. Ecrire un programme en Turbo-Pascal qui lise un entier N et un réel β et qui affiche, en sortie, les N premiers termes de la suite V.

2.15 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 $b_{n+1} = \dfrac{b_n}{n+1}$
Ecrire un programme en Turbo Pascal qui calcule et affiche les n premiers termes de chacune des suites (a_n) et (b_n) pour une valeur de n entrée par l'utilisateur.

3 Sommes, produit (Accumulateurs)

3.1 Mathématiques : les sommes sont utilisée pour

- Calculer des moyennes, (expériences probabilitstes)
- Calculer des valeurs approchées d'intégrales (Sommes de Rieman)
- Comme compteur
- Calculs de puissances, ou de factorielles.

3.2 Algorithme :

On utilise la relation de récurrence $\sum_{k=1}^{n+1} u_k = \sum_{k=1}^{n} u_k + u_{n+1}$
`S:=0;` et on répète `S:=S+u` où `u` devra contenir le terme suivant.
Si le terme suivant se calcule lui même par récurrence :
`S:=u0;` puis répète {`u:=f(u);S:=S+u;`}

4 Probabilités

4.1 Les outils :

4.1.1 Tirage au hasard

`randomize;` initialise le générateur de nombre aléatoire : à n'utiliser qu'une seule fois en début de programme.
`random(n);` donne équiprobablement les entiers de $[[0, n-1]]$.
Pour avoir un entier de $[[1, n]]$: `random(n)+1.`
`random;` donne équiproablement les réels de $[0, 1[$ c'est à dire qu'avec `r:=random;`
P(a<r and r<b)=b-a si 0≤a≤b≤1
On simule un tirage par un `if (a<r and r<b)` de même probabilité et incompatibilité.

4.1.2 Trirages sans remise

Il faut connaitre le nombre de boules de chaque type restant.
Il faut un compteur pour chaque type de boule.
Exemple :
On effectue le tirage sans remise de 50 boules parmi 2 boules rouges et 100 boules vertes.
Quelle est le nombre de rouges et de vertes obtenues.

```
function boule(r,v:integer):char;
begin if random<r/(r+v) then boule='R' else boule='V';
end;
```

donnera R avec une probabilité de $\frac{r}{r+v}$et V avec une probabiltié $\frac{v}{r+v}$
`r:=2;v:=100; ... if boule(r,v)='R' then r:=r-1 else v:=v-1;`

4.1.3 Nombre de

On utilise un compteur qui s'incrémente sous condition : `C:=0;...; if D=6 then C:=C+1;`
ou un tableau de compteurs
Exemple :
compter le nombre de 1, 2 ,...,6 obtenus dans des lancers de dé.
`for i:=1 to 6 do C[i]:=0; ... D:=random(5)+1; C[D]:=C[D]+1;`

4.1.4 Moyenne

On calcule la somme des résultat et on divise par le nombre de résultats.
Il faut un accumulateur pour la somme et un compteur pour le nombre de résultats, s'il n'est pas connu d'avance.
Exemple :
La moyenne des résultats impairs (`odd`) obtenus en 100 tirages:

```
S:=0;C:=0;randomize;
for i:=1 to 100 do
begin
     D:=random(5)+1;
     if odd(D) then begin c:=c+1;S:=S+D end;
end;
if C<>0 then writelen(' moyenne des impairs :';S/C) else writeln('pas d''impairs');
```

4.1.5 Maximum ou minimum

Pour déterminer le maximum ou le minimum, on initialise avec le premier, et on met à jour à chaque nouveau.

Exemple :

max et min en 10 lancers de dés :

```
randomize;
D:=random(5)+1;
max:=D;min:=D;
for i:=2 to 10 do
begin
     D:=random(5)+1;
     if D>max then max:=D;
     if D<min then min:=D;
end;
writelen('maximum :  '  ,max,' minimum :  ',min);
```

4.2 EDHEC 2007

On lance une pièce équilibrée et on note Z la variable aléatoire égale au rang du lancer où l'on obtient le premier "pile".
Après cette série de lancers, si Z a pris la valeur k ($k \in \mathbb{N}^*$), on remplit une urne de k boules numérotées 1, 2, $\cdots$, k, puis on extrait au hasard une boule de cette urne.
On note X la variable aléatoire égale au numéro de la boule tirée après la procédure décrite ci-dessus.
On décide de coder l'événement ≪obtenir un "pile"≫ par 1 et l'événement ≪obtenir un "face"≫ par 0.
On rappelle que la fonction `random` renvoie, pour un argument k de type `integer` (où k désigne un entier supérieur ou égal à 1) un entier aléatoire compris entre 0 et $k-1$.

1. Compléter le programme suivant pour qu'il affiche la valeur prise par Z lors de la première partie de l'expérience décrite ci-dessus.

```
Program edhec_2007;
Var z,hasard:integer;
begin
     randomize; z:=0;
     repeat
          z:=.........; hasard:=.........; until (hasard=1);
     writeln(z);
end.
```

2. Quelle instruction faut-il rajouter avant la dernière ligne de ce programme pour qu'il simule l'expérience aléatoire décrite dans ce problème et affiche la valeur prise par la variable aléatoire X ?

4.3 EML 2007

On a montré que $Y+1$ suit une loi géométrique de paramètre e^{-1}.
Recopier et compléter le programme ci-dessous pour qu'il simule la variable aléatoire Y
`program eml2007;`

1. **a)**
```
var y:integer; u:real;
begin
     randomize;
     u:=random; y:=···;
     while...  do
     ...  ...  ...
     writeln('y vaut ', y);
end.
```

4.4 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 dont l'entête est

`function aleat(b,r:real):integer;`

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

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

```
begin
     writelen('proportion de rouges et de blanches ?');
     readln(r,b);
     randomize;
     x:=aleat(r,b);
     k:=...;
     repeat k:=k+1 ; y:=...  until ....;
     writeln( 'il a fallut ',k,'lancers');
end.
```

4.5 EDHEC 2004

On désigne par n un entier naturel supérieur ou égal à 2.
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.
On rappelle que l'instruction `random(2)` renvoie un nombre au hasard parmi les nombres 0 et 1.
Recopier et compléter le programme suivant pour qu'il simule l'expérience décrite ci-dessus, l'entier n étant entré au clavier par l'utilisateur (pile sera codé par le nombre 1 et face par 0).

```
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 ..........  ;
     until (lancer = 1) or (..........)  ;
     Writeln (z) ;
end.
```

4.6 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;

On rappelle que dans un programme PASCAL, l'instruction " `r :=RANDOM(2)`" a pour effet de donner aléatoirement à la variable r la valeur 0 ou 1, ces deux valeurs étant équiprobables.
On considère la procédure PASCAL suivante :

```
PROCEDURE Quigagne;
VAR x,y,r,k :INTEGER;
BEGIN
     x :=0; y :=0;k :=0;
     WHILE x<3 AND y<3 DO
     BEGIN
          k :=k+1; r :=RANDOM 2;
          IF r=1 THEN
          BEGIN
               IF x>=1 THEN x :=2 ELSE x :=1;
               IF y>=1 THEN y :=y+1;
          END
          ELSE
          BEGIN
               IF x=2 THEN x :=3
               ELSE x :=0;
               y :=1;
          END;
     END;
     IF x=3 THEN WRITE (' ..... ') ELSE WRITE(' .....');}
END;
```

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) 1, 1, 1, 1, 0

b) 1, 0, 1, 0, 0, 0, 1, 1

c) 0, 1, 0, 1, 0, 1, 1

2. Que représente la dernière valeur prise dans la procédure par la variable k et quels textes pourrait-on substituer aux pointillés de la dernière instruction?
Qu'afficherait alors l'ordinateur dans les trois exemples de la question précédente ?

4.7 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 $T_{\ 12}$.

On rappelle que `random(6)` fournit un entier aléatoire parmi 0, 1, 2, 3, 4, 5 .

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

4.8 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 $\{1, 2\}$

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

On appelle Y le contenu de y après exécution du programme ESSEC2003.

Donner la loi de Y, calculer et son espérance $E(Y)$

4.9 Ancien : Inégalité des accroissements finis

Soit $f(x) = \frac{1}{2}e^{-x}$ et $u_0 = 0$ et pour tout entier n, $u_{n+1} = f(u_n)$

1. Etude de f

a) Déterminer le sens de variations de f.

b) Montrer que si $x \in [0,1]$ alors $f(x) \in [0,1]$

c) Monter que l'équation $f(x) = x$ a une unique solution α et que $\alpha \in [0,1]$

d) Montrer que $|f'| \leq 0,5$ sur $[0,1]$

2. Convergence de u

a) Montrer que pour tout entier n, $u_n \in [0,1]$

b) En déduire que pour tout entier n, $|u_{n+1} - \alpha| \leq 0,5\,|u_n - \alpha|$ puis que $|u_n - \alpha| \leq 0,5^n$ et enfin la limite de la suite u_n

c) A quelle condition u_n donne-t-elle une valeur approchée de α à ε près ?

3. Programmation du calcul de la valeur approchée de α

Ecrire un programme qui calcule u_n jusqu'à ce que $0,5^n \leq 10^{-3}$ et affiche une valeur approchée de α à 10^{-3} près.

(problème : il n'y a pas de fonciton puissance en PASCAL)

4.10 Suites adjacentes

Soit $u_0 = 0$ et pour tout entier n : $u_{n+1} = e^{-u_n}$. Soit $f(x) = e^{-x}$ et
$g(x) = e^{-e^{-x}} = f(f(x)) = f \circ f(x)$

1. Etude de f et de g

a) Déterminer le sens de variations de f et montrer que l'éqation $f(x) = x$ a une unique solution α et que $\alpha \in [0,1]$

b) Montrer que α est solution de $g(x) = x$

c) Soit $h(x) = g(x) - x$.
En calculant h'', montrer que $g(x) = x$ n'a pas d'autre solution que α, et que si $x \leq \alpha$ alors $x \leq g(x)$

2. Etude de la suite v : $v_n = u_{2n}$

a) Montrer que pour tout entier n, $v_{n+1} = g(v_n)$

b) Montrer que pour tout entier n, $v_n \leq \alpha$

c) En déduire que v est croissante et converge vers α.

3. Etude de $w_n = u_{2n+1}$

a) Montrer que pour tout entier n, $w_n = f(v_n)$

b) Déduire de ce qui précède que w est décroissante et converge vers α

4. Programmation du calcul d'une valeur approchée de α :

a) Déterminer pour que v soit une valeur approchée de α à ε près.

b) Ecrire un programme qui calcule et affiche une valeur approchée de α à 10^{-3} près.

4.11 Sommes et puissances

On note pour tout entier n : $S_n(x) = \sum_{k=0}^{n} \frac{x^k}{k!}$

1. Limite de $(S_n(x))_{n\in\mathbb{N}}$

On pose pour tout entier n :

$$f_n(x) = e^x - \sum_{k=0}^{n} \frac{x^k}{k!} \quad \text{et} \quad g_n(x) = e^x - \sum_{k=0}^{n} \frac{x^k}{k!} - (e-1)\frac{x^n}{n!} = f_n(x) - (e-1)\frac{x^n}{n!}$$

a) Montrer que pour tout entier n, $\quad f'_{n+1}(x) = f_n(x)$ et en déduire par récurrence sur n que pour tout entier n et tout $x \in [0,1]$, $\quad f_n(x) \geq 0$

b) Montrer de même que pour tout entier n et tout $x \in [0,1]$, $\quad g_n(x) \leq 0$

et donc que $f_n(x) \leq (e-1)\frac{x^n}{n!}$

c) En déduire que pour tout $x \in [0,1]$:

$$0 \leq e^x - S_n(x) \leq \frac{e-1}{n!}x^n \leq 2\frac{x^n}{n!}$$

d) En déduire la limite de $S_n(x)$ quand n tend vers $+\infty$.

2. L'objet de cette partie est d'écrire un programme en PASCAL permettant de calculer une valeur approchée de e^x avec la précision voulue pour $x \in [0,1]$.

Remarque : dans le programme qui suit, la difficulté est de bien identifier si les variables contiennent les valeurs suivantes ou précédentes (indice k ou $k+1$)

a) A quelle condition $S_n(x)$ donne-t-elle une valeur approchée de e^x à ε près ?

Pour ce faire, on affectera les sommes $S_n(x)$ à une variable S, les valeurs de k à une variable K, les valeurs de $x^k/k!$ à une variable F et la précision voulue à une variable *eps*

b) Quelles sont pour $n = 0$ les valeurs de F, et de S ?

c) Comment obtient-on $x^{k+1}/(k+1)!$ à partir de $x^k/k!$?

Comment obtient-on la valeur suivante de F à partir de sa valeur précédente, si K contient la valeur suivante $(k+1)$?

d) Comment obtient-on la valeur suivante de S à partir de la précédente si F contient la valeur suivante ?

e) Ecrire un programme qui demande une précision puis calcule et affiche une valeur approchée de e^x à cette précision, ainsi que l'écart avec la valeyur exacte (`exp(x)`)

5 Ancien : Probabilité

5.1 Simulation du hasard

Pacal dispose de la fonction Random pour simuler le hasard. Elle doit être initialisée par `Randomize`(une seule fois)

- Pour `n` un entier, `random(n)` renvoie un entier de l'intervalle $[[0, n-1]]$, tous les entiers étant équiprobables.

  Par exemple, la fonction définie par :

  ```
  function de:integer;
  begin
  de:=random(6)+1
  end;
  ```

  simulera un dé (les entiers de 0+1 à 6-1+1 seront équiporbables)

- `random` sans argument renvoie un réel de l'intervalle $[0, 1[$ la densité étant équirépartie (la probabiltié est proportinnelle à la longueur de l'intervalle dans lequel on recherche une valeur)

  Par exemple la fonction définie par :

  ```
  function PF(p:real):char;
  begin
  if random<p then PF:='P' else PF:='F';
  end;
  ```

  simulera un lancer de pièce truquée, la probabilité de 'P' étant p.

5.2 Nombre de succès, moyenne, max min

1. On lance 100 fois une pièce truquée dont la probabiltié de donner Pile est 0,3.

a) Quelle est la loi du nombre Pile obtenu ?
Quelle est le nombre de Pile obtenus en moyenne.

b) Ecrire un programme qui simule
une telle série de lancers et qui affiche

- le nombre de Pile obtenus et le nombre moyen de pile par lancer.
- Plus compliqué : la longueur de la plus longue liste de Pile obtenus.

2. On lance cette pièce jusqu'à obtenir Pile.

a) Quelle est la loi du rang du premier Pile et sa moyenne ?

b) Ecrire un programme qui simule le lancer de la pièce jusqu'à obtenir Pile et qui affiche le nombre de lancers effectués.

c) Ecrire un programme qui simule 1000 séries lancer de la pièce jusqu'à obtenir Pile et qui affiche

- le nombre moyen de lancers effectués
- le temps d'attente maximum et minimum.

5.3 trirages sans remise

Pour modéliser des tirages sans remise parmi 2 boules rouges et 100 boules vertes, il faut connaitre lors de chaque tirage le nombre de boules rouges et vertes restantes.

1. Etude théorique :

a) Déterminer la loi de X le rang d'apparition de la première boule rouge et déterminer l'espérance de X.

b) Soit Y le rang d'apparition de la seconde.

Justifier $(Y = n) = \bigcup_{k=0}^{n-1} (V_1 \cap \cdots \cap V_{k-1} \cap R_k \cap V_{k+1} \cap \cdots \cap V_{n-1}) \cap R_n$ et en déduire la loi de Y et calculer son espérance.

2. Programmation :

a) S'il y a r rouges et v vertes dans l'urnes quelle est la probabilité d'obtenir une boule rouge? Comment ontenir avec Random une telle probablité ?

b) Ecrire un programme qui simule l'extraction des boules de l'urne jusqu'à ce qu'elle soit vide (combien de tirages?) et qui affiche le rang d'apparition de la première boule rouge et celui de la deuxième.

c) Ecrire un programme qui recommence 100 fois ces tirages et qui affiche le rang moyen d'apparition de la première et de la seconde rouge

5.4 Plusieurs compteurs

Pour compter le nombre de sortie de chaque face d'un dé, on utilsie un tableau de compteur
`var Cpt:array[1..6]of integer;`
Si le résultat du dé est dans D, on augmente de le compteur D : `Cpt[D]:=Cpt[D]+1`

1. On lance 100 puis 1000 puis 10000 fois un dé.

Ecrire un programme qui totalise les sorties de chacunes des faces et qui affiche

- le nombre moyen de sortie de chacune des faces. (on peut afficher tous les compteurs avec une seule instruction :

  `for i:=1 to 6 do write(cpt[i]/1000);writeln;`
- l'écart par rapport à la moyenne théorique

2. On lance trois dés.

a) Quelle est la probabilité d'obtenir un triple 6 ? Un 421 (dans le désordre)

b) Comment savoir si l'on a eu un "421" (dans le désordre) ou un "666" ?

c) Ecrire un programme qui simule trois lancers de dés et qui détermine si l'on a eu 421 et si l'on a eu triple 6. De même avec 1000 fois trois lancers et en affichant le nombre moyens de 421 et de 666 obtenus.

d) Ecrire un programme qui simule trois lancers de dés jusqu'à obtenir 421 et qui affiche le nombre de triples lancers.