NF01 – Printemps 2018

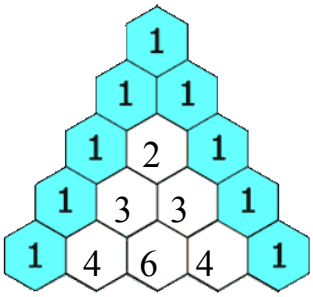
Examen Final

ATTENTION !
Utilisez quatre copies séparées, une par partie.

Seul le polycopié de cours est autorisé.

1ère Partie : Tableaux *(5 points)*

Le triangle de Pascal est le triangle dont la ligne d'indice n (n = 0, 1, 2...) donne les coefficients binomiaux C_n^p pour p = 0, 1, 2..., n. (voir figure ci-dessous).



1								
1	1							
1	2	1						
1	3	3	1					
1	4	6	4	1				
1	5	10	10	5	1			
1	6	15	20	15	6	1		
1	7	21	35	35	21	7	1	
1	8	28	56	70	56	28	8	1

Pour avoir un terme de la ligne suivante, on prend le terme juste au-dessus, et on lui additionne celui qui est juste avant (0 s'il n'y a rien).

a. Définir une procédure `calculerTriangle(m,n)` qui remplit une matrice `m` avec un triangle de Pascal d'indice `n` construit suivant le principe décrit ci-dessus. L'indice `n` représente la dimension effective de la matrice avec `n` ≤ `MAX` et `MAX = 20`. La partie triangulaire supérieure doit être mise à 0.

b. Définir une procédure `afficherTriangle(m,n)` qui utilise la matrice `m` de dimension `n` pour afficher seulement la partie triangulaire inférieure (comme dans le tableau ci-dessus).

c. Ecrire une fonction `symetrique(t,n)` permettant de vérifier qu'un tableau `t` de `n` nombres entiers est constitué symétriquement de nombres identiques.

Exemple :

1									MAX
14	9	25	25	9	14				

d. Donner une expression permettant de vérifier que les éléments de la ligne `i` de la matrice du triangle de Pascal sont symétriques.

{ ---------- pensez à changer de copie ---------- }

2ème Partie : Récursivité *(5 points)*

Un principe connu depuis fort longtemps permet d'effectuer la multiplication de deux entiers naturels a et b. Il est défini comme suit :

- si b vaut 0 alors : a × b est égal à 0
- si b est impair alors : a × b est égal à a + a × (b – 1)
- si b est pair alors : a × b est égal à (a + a) × (b / 2)

1. Ecrire une fonction <u>récursive</u> nommée *produit* qui applique ce principe pour multiplier deux nombres entiers a et b.
2. Faire une simulation avec a = 4, b = 3.

{ ---------- pensez à changer de copie ---------- }

3ème Partie : Fichiers *(5 points)*

On considère le programme incomplet suivant :

```
program MesPhotos;

const
    N = 100;

type
    ListTag = array [1..N] of string;
    Image = record
                nomFichier : string; //nom du fichier de la photo
                tags : listTags ; //tags pour commenter la photo
            end;
    FichierImages = file of Image;

var
   fApril, fRaphael : FichierImages;

begin
    assign(fRaphael, 'AlbumRaphael.txt');
    rewrite(fRaphael);

    assign(fApril, 'AlbumApril.txt');
    rewrite(fApril);
    …
    close(fRaphael);
    close(fApril);
end.
```

L'utilisateur peut donc associer à chaque photo d'un album des tags comme le nom des personnes se trouvant sur la photo (e.g. Leonardo, Raphaelo), l'événement associé à la photo (e.g. PizzaParty, SoireePic), etc.

1) Ecrire une procédure `saisie` en Pascal qui permette de saisir autant de photos que l'utilisateur le souhaite dans un fichier de type `FichierImages` passé en paramètre. La

procédure demande à l'utilisateur le nombre de photos qu'il souhaite saisir. On veillera à ce que ce nombre ne dépasse pas `N`[1]. Pour chaque photo, la procédure demande : le nom du fichier de la photo, le nombre de tags et les tags associés à cette photo *(2 points)*.

Par exemple, l'appel `Saisie(fRaphael)` pourrait permettre de saisir les informations suivantes dans le fichier :

```
20180625_1.jpeg, tags : Leonardo, Michelangelo, PizzaParty,
20180625_2.jpeg, tags : Michelangelo, PizzaParty,
20180625_3.jpeg, tags : Leonardo, Donatello, PicParty, Shredder,
20180625_4.jpeg, tags : Donatello, PizzaParty,
```

2) Ecrire une fonction `recherchePhoto` en Pascal qui demande à l'utilisateur le nom du fichier de la photo qu'il recherche et qui renvoie la position à laquelle se trouve cette photo dans le fichier `fImages` passé en paramètre, et -1 si la photo n'existe pas *(1 point)*.

Par exemple, l'appel `i:= recherchePhoto(fRaphael)` afficherait :

```
***Quel fichier recherchez-vous :
20180625_2.jpeg
--- Nom trouvé en position : 1
i sera égal à 1 dans ce cas
```

3) Ecrire une procédure `rechercheTag` en Pascal qui demande le tag recherché et qui affiche toutes les photos du fichier `fImages` passé en paramètre portant ce tag. La procédure affiche aussi le nombre total de photos possédant ce tag *(2 points)*.

Par exemple, l'appel `rechercheTag(fRaphael)` afficherait :

```
***Quel tag recherchez-vous :
PizzaParty
--- Tag trouvé dans photo 20180625_1.jpeg
--- Tag trouvé dans photo 20180625_2.jpeg
--- Tag trouvé dans photo 20180625_4.jpeg
Nombre de photos trouvées : 3
```

{ ---------- pensez à changer de copie ---------- }

4ème Partie : Chaines de caractères *(5 points)*

On veut écrire un programme en Pascal qui permette d'extraire des chiffres présents dans un texte.

On considèrera qu'un texte est un ensemble de lignes.

1) Ecrire une procédure permettant la saisie du texte, et rangeant ce texte dans un tableau de lignes (avec 100 lignes au maximum).

   Chaque élément du tableau représentera une ligne de caractères, entrée au clavier par l'utilisateur, jusqu'au return (pour indiquer la fin de ligne). Cette procédure doit retourner le tableau ainsi composé.

2) Proposez une fonction qui, lors de son appel, reçoit une chaîne de caractères et retourne le nombre extrait de cette chaîne de caractères.

   Exemple : si la chaîne de caractères est "ser23df91" la fonction retourne le nombre entier 2391.

[1] Pensez à la façon de gérer par la suite de manière intelligente le nombre de tags

3) Ecrire une procédure à deux paramètres permettant de transformer le texte saisi précédemment (à la question 1) en une série de nombres, en utilisant la fonction de la question 2.

   Ces nombres seront rangés dans un tableau d'entiers dont chaque élément correspondra à une ligne du texte. Les deux paramètres sont le tableau de lignes et le tableau d'entiers.

4) Ecrire une fonction qui calcule la moyenne des nombres obtenus à la question 3. Cette fonction reçoit le tableau avec les nombres et retourne la moyenne.

5) Ecrire une procédure permettant d'afficher tous les nombres ainsi trouvés, pour le texte saisi par l'utilisateur, ligne par ligne ainsi que la moyenne calculée à la question 4.