Exercice 1

Le Crible d'Eratostène est un algorithme classique pour énumérer les nombres premiers. Supposons que nous voulions trouver les nombres premiers plus petits que 10. Commençons par écrire les nombres de 2 à 10 : 2 3 4 5 6 7 8 9 10
Nous sélectionnons le plus petit (c'est-à-dire : 2), certains qu'il est premier, et nous retirons tous ses multiples. Après cette première étape, nous avons : 2 3 5 7 9.
A la seconde étape, nous savons que 3 est premier. Nous retirons tous ses multiples. Après cette seconde étape, nous avons : 2 3 5 7.
A la troisième étape, nous savons que 5 est premier. Nous retirons tous ses multiples. Après cette étape, nous avons : 2 3 5 7.
A la quatrième étape, nous savons que 7 est premier. Nous retirons tous ses multiples. Après cette étape, nous avons trouvé les nombres premiers plus petits que 10 :

2 3 5 7

Après l'initialisation d'un tableau avec les entiers suivants : 2…1000, suivre le principe énoncé ci-dessus pour afficher les nombres premiers plus petits que 1000.

Exercice 2

Le but de cet exercice est de construire le triangle de Pascal (voir le tableau ci-dessous) pour un nombre de lignes donné. Le nombre de colonnes est égal au nombre de lignes. On envisage ici deux méthodes : la première utilise un tableau à deux dimensions et la deuxième utilise un tableau à une dimension.

1. Écrire l'algorithme de construction du triangle de pascal pour un nombre de lignes donné. La taille maximale de la matrice est de [100,100].
2. Afficher les éléments de cette matrice. Afficher uniquement les éléments du tringle.
3. Écrire une action qui, à partir de la ligne *n* du triangle de Pascal stockée dans un tableau à une dimension de taille 100, calcule les valeurs de la ligne *n+1* en effectuant une mise à jour du tableau sans utiliser de tableau intermédiaire. Le tableau est supposé d'une taille suffisante (n<100). Le prototype de cette fonction est le suivant :

Procedure constLigneSuivante(Tab, n) // n représente le n° de la ligne

Exemple pour n = 4 et avec un tableau de taille 10 :

1	3	3	1	0	0	0	0	0	0

Tab **avant** l'appel de la procédure : n=4

1	4	6	4	1	0	0	0	0	0

Tab **après** l'appel de la procédure : n=5

4. On souhaite imprimer, parmi les 100 premières lignes du triangle de Pascal, toutes celles qui ne contiennent que des valeurs strictement inférieures à un nombre strictement positif donné *nb>1*. Pour résoudre ce problème, on vous demande d'utiliser l'action de la question précédente pour calculer et afficher une à une les lignes du triangle de Pascal jusqu'à la première ligne contenant une valeur supérieure ou égale au nombre *nb* donné strictement positif. Cette ligne ne sera pas affichée. Si on atteint la ligne 100 sans avoir trouvé, on s'arrête. La valeur de *nb* est saisie au clavier. On choisit de ne pas afficher les 0 présents dans une ligne. Écrire le programme correspondant.

1	0	0	0	0

1	1	0	0	0
1	2	1	0	0
1	3	3	1	0
1	4	6	4	1

LISTES

Exercice 1

1. Initialiser une liste simplement chaînée avec les entiers suivants : 2…1000.
2. En suivant le principe donné par le Crible d'Eratostène, construire la liste des nombres premiers plus petits que 1000. Afficher cette liste.

Exercice 2

Ecrire une fonction qui interclasse 2 listes d'entiers triées par ordre croissant (sans perdre les listes initiales).

Exercice 3

À partir d'une liste d'entier quelconque (non triée) on veut créer une liste triée par ordre croissant. La procédure à suivre est la suivante :

1. Donner le type de cette liste
2. Créer la liste d'origine en utilisant la fonction suivante:
   Fonction Creation_liste(E/ Tete1 : Liste) : Liste

Tant qu'il reste des éléments dans cette liste (liste d'origine)

Faire

3. Chercher le maximum :
   Fonction Chercher_Max(E/ Tete1 : Liste) : entier
4. Libérer ce maximum de cette liste :
   Fonction Liberer_Max_Element (E/ Tete1 : Liste, E/ max : entier) : Liste
5. Créer un nouvel élément de la liste résultat :
   Fonction Creation_Element_Liste(E/ Tete2 : Liste, E/max : entier) : Liste

Fait

6. Afficher la liste résultat : Procedure Affiche_Liste(E/ Tete2 : Liste)

Exercice 3.1 (même exercice mais dans ce cas pas de libération ni allocation d'espace)

Tant qu'il reste des éléments dans cette liste (liste d'origine)

Faire

2. Chercher le maximum :
   Fonction Chercher_Max(E/ Tete1 : Liste) : Liste //la fonction retourne l'adresse du Max
3. Créer un nouvel élément de la liste résultat :
   Procedure Creation_Element_Liste(ES/ Tete1, Tete2 : Liste, E/Pmax : Liste)

Fait

4. Afficher la liste résultat : Procedure Affiche_Liste(E/ Tete : Liste)

Exercice 4

1. Ecrire une action paramétrée qui à partir d'une chaîne de caractère construit une liste circulaire composée uniquement des lettres de cette chaîne (voir figure 1).

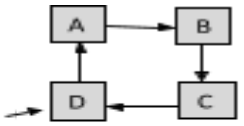

Figure 1 : liste circulaire correspondant à la chaine ch="ABCD"

2. En déplaçant la queue de la liste circulaire (voir la figure 2), écrire une action paramétrée qui à partir de la liste circulaire de caractère, construit toutes les chaînes possibles (sauf la chaîne initiale) et sauvegarde ces mots dans une liste simplement chaînée.

Remarque : traiter les cas particuliers suivants : ch=''ABAB'', ch= ''A'', ch=''AAA'', …

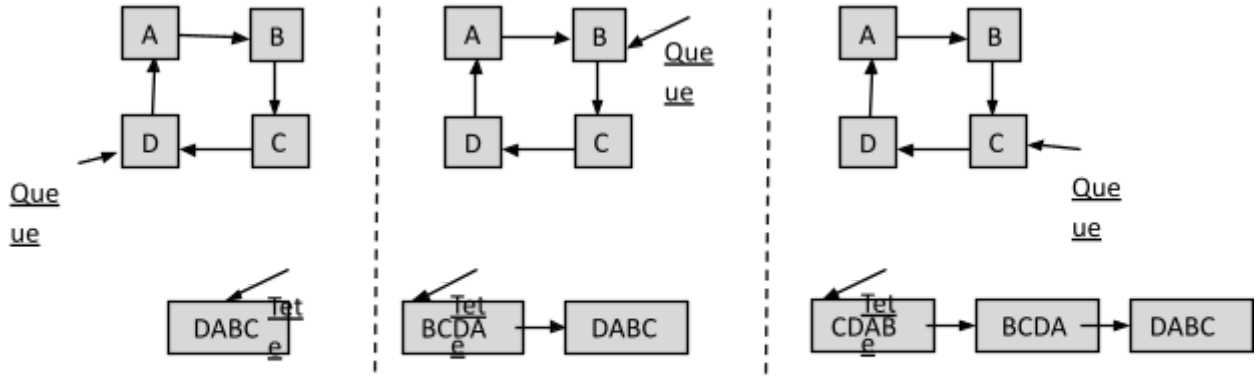

Figure 2 : liste simplement chaînée correspondant à toutes les chaînes possibles en déplaçant la queue de la liste circulaire représentant la chaîne ch=''ABCD''

3. Pour un ensemble de N chaînes de caractères, écrire l'algorithme principal qui:
   - construit la liste chaînée correspondant à N chaînes de caractères
   - pour chaque élément de cette liste, construit les chaînes possibles (voir la figure 3)

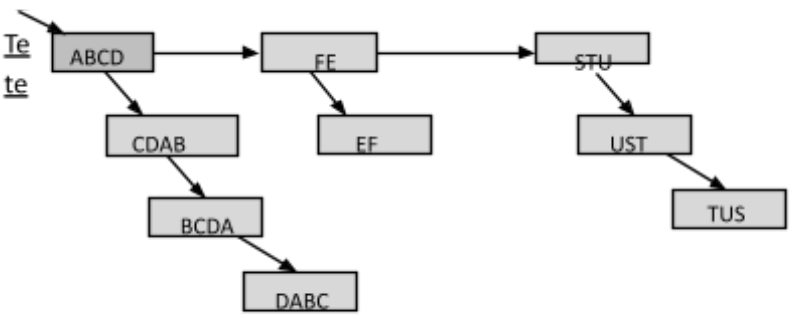

Figure 3: liste de N=3 chaînes de caractère où chaque chaîne est attachée une liste représentant toutes les combinaisons possibles (d'après l'énoncé donnée par la question 2)

Exercice 5

A partir d'un nombre entier positif x, on veut générer tous les nombres possibles en suivant le principe suivant : pour chaque chiffre constituant ce nombre, on génère deux nombres, en allant vers la droite puis vers la gauche de ce chiffre. Exemple : pour le nombre 216 et en commençant par le chiffre 6 on génère : 621 ($6*10^2+2*10^1+1*10^0$) puis 612 ($6*10^2+1*10^1+2*10^0$). Pour le chiffre 2 on génère : 216 puis 261. Pour le chiffre 1 on génère : 162 puis 126. Ce même principe est appliqué pour l'ensemble: {15, 7, 191} illustré par la figure 1. Notons qu'on ne tient pas compte des doublons et du nombre initial s'il est généré. Voir le cas de 191 par exemple. En appliquant ce principe, on vous demande :

4. Donner le(s) type(s) de la structure représentée par la figure 1.
5. Ecrire une action paramétrée qui construit une liste **circulaire bidirectionnelle** composée uniquement des chiffres d'un nombre entier positif (voir la figure 2).
6. En utilisant **uniquement** la queue de la liste circulaire bidirectionnelle comme paramètre d'une action paramétrée, construisez la liste de tous les nombres possibles en suivant le principe donné en haut de cet exercice (voir la figure 3).

Remarque : à exclure les doublons et le nombre initial voir par exemple pour le nombre 191.

7. Dans l'algorithme principal, on vous demande de:
   - Créer une liste simplement chaînée pour les nombres suivant : 15, 2016, 7, 191
   - Créer la liste des nombres possibles pour chaque élément de la liste initiale.

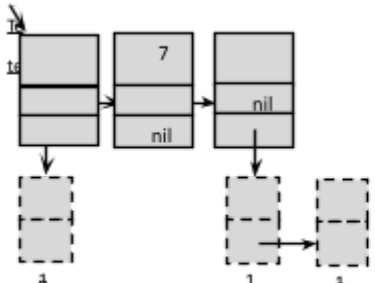

Figure 1: liste de N entier (15, 7, 191). Pour chaque chiffre d'un nombre, on génère deux nombres en en suivant le principe donné ci-dessus

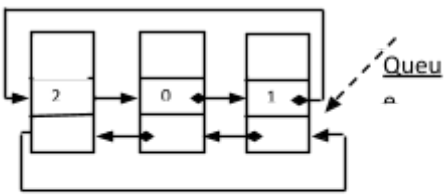

Figure 2 : liste circulaire bidirectionnelle correspondant au nombre 201

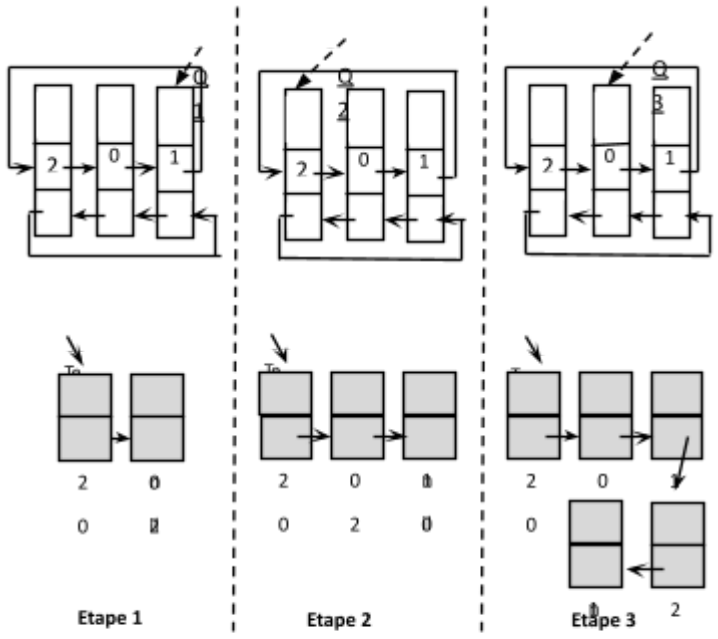

Figure 3 : étapes de construction de tous les nombres possible à partir d'une liste circulaire bidirectionnelle représentant le nombre 201