Feuille n°2 : Itération - Les boucles tant que, Répéter et Pour

Partie 1

```
Algorithme DeuxChiffres
    const (STOP : entier) = 99999
    var  uneVal,  : entier
    début
        ecrire ('Entrer  un nombre' , STOP,  ' pour finir. ')
        uneVal  <- lire ()
        tant que  uneVal ≠ STOP  faire
            si uneVal ≥  10 et uneVal < 100
                alors ecrire ( uneVal, 'est un nombre à deux chiffres. ')
            finsi
            ecrire ('Entrer  un autre  nombre, ' , STOP,  ' pour finir. ')
            uneVal <- lire()
        fintantQue
        ecrire ('fin de l'algorithme ')
    fin
```

a) Que fait cet algorithme ? Cet algorithme attend des données de la part de son utilisateur. Quels affichages obtenez -vous si vous lui fournissez les données suivantes : 23 142 99999.
b) Commentez cet algorithme afin d'identifier clairement les différentes parties du traitement itératif effectué par la boucle tant que.
c) Pourquoi la boucle tant que est-elle préférable ici à la boucle Répéter et à la boucle Pour ?

Partie 2 : Complétez les algorithmes suivants afin qu'ils effectuent le traitement itératif décrit.
Justifiez clairement votre choix de boucle.

a) Algorithme Factorielle

```
    //Cet algorithme calcule la factorielle d'un nombre fourni  par l'utilisateur
var  val,  fact: entier
début
    ecrire( 'Entrez un nombre entier ')
    val <- lire ()
    …
    //affichage du résultat
    ecrire( ' La factorielle de ', val, ' est  ',  fact)
fin
```

b) Algorithme Bonjour

```
//Cet algorithme simule un bonjour poli et personnalisé}
    variables réponse : entier
        début
        // Demander à l'utilisateur de taper 1 si elle est une étudiante,
        //et 2 si il  est  un  étudiant, et réitérer la demande tant que les consignes ne sont pas suivies
            …
        //affichage du résultat}
        si réponse = 1
            alors ecrire ('Bonjour Mademoiselle ')
            sinon ecrire (' Bonjour Monsieur ')
        finsi
    fin
```

c) Algorithme AffichePhrase

```
 //Cet algorithme saisit et affiche au fur et à mesure les caractères fournis par l'utilisateur  jusqu'au //premier point
```

rencontré. En fin de traitement, il affiche le nombre de caractères saisis.

```
constante (STOP : caractère) = #.
var unCar : caractère
début
        …
        //présentation des résultats
        ecrire ('Le nombre de caractère saisis est ' , cpt)
fin
```

Attention : le point final est la valeur d'arrêt et ne doit donc pas être affiché, ni comptabilisé.

Partie 3 :

a) Modifiez l'algorithme AffichePhrase pour :
- qu'il n'affiche pas les blancs (espaces). L'algorithme devra afficher, en fin de traitement, le nombre de caractères blancs éliminés.
- et qu'il saisisse au maximum 100 caractères (pensez aux constantes).

L'algorithme devra également indiquer, en fin de traitement, la raison de la sortie de boucle : a-t-on saisi la valeur d'arrêt ou bien a-t-on dépassé le maximum de valeurs autorisé ?
Justifiez la boucle choisie.

b) On veut trouver la plus grande valeur dans une suite de valeurs saisies par l'utilisateur.
Ecrire un algorithme qui
- demande à l'utilisateur le nombre nbVal de valeurs entières à saisir,
- saisit les nbVal valeurs,
- affiche la plus grande valeur ainsi que son rang dans la suite.

Par exemple, si la suite de valeurs est 45 –6 35 11 –345 462 –24
alors le programme affichera : La plus grande valeur est 462, c'est la 6 ème valeur de la suite.
Justifiez la boucle choisie.

c) On veut saisir une valeur dans un intervalle donné. Pour cela :

a. Ecrire un algorithme qui saisit une valeur de référence valdébut, et puis saisit une suite de nombres et s'arrête dès que l'on saisit une valeur qui soit supérieure à valdébut. A l'issue du traitement, l'algorithme doit afficher la valeur qui a provoqué l'arrêt, ainsi que son rang. Justifiez la boucle choisie.

b. Modifiez votre algorithme pour qu'il saisisse deux entiers référence valdébut et valfin délimitant un intervalle donné (vous supposerez que valdébut est plus petit que valfin), et qu'il arrête la boucle de saisie dès que l'on saisit une valeur qui soit dans cet intervalle. A l'issue du traitement, l'algorithme doit afficher la valeur qui a provoqué l'arrêt ainsi que son rang.

c. On ne suppose plus que valdébut est plus petit que valfin. Améliorez votre algorithme pour permettre à l'utilisateur de ressaisir les deux valeurs de référence tant que la première n'est pas inférieure à la deuxième.

Partie 4 : On veut pouvoir compter le nombre de doublets (c'est à dire le nombre de paires de lettres doubles) dans une phrase en français. Par exemple, la phrase suivante contient 5 doublets:

Oolithe est un mot bizarre, assez connu des assidus géologues.

a. Ecrire un algorithme qui saisit une phrase (c'est à dire une suite de caractères) terminée par un point, et qui compte le nombre de doublets. (Vous pouvez supposer qu'il n'y a pas de lettres triples).
Vérifiez que votre algorithme donne le résultat attendu pour tous les cas particuliers. Pensez au cas de la phrase « vide », de la phrase à un seul caractère, de la phrase avec un doublet en début, en fin ou au milieu.
Conseil : Utilisez deux variables carCour et carPréc pour disposer en même temps du caractère qui vient d'être saisi (carCour) et du caractère précédent (carPréc).

b. Modifiez votre algorithme pour qu'il ne compte que les doubles 's'.

c. On ne suppose plus que des suites de trois lettres soient interdites. Comment interviennent-elles dans votre algorithme ? Comment devrait-on le modifier pour les compter ? Voici un exemple d'une phrase contenant deux triplets :

Un stress stupéfiant gagne le zoo oriental.