

Programmation Impérative

1ère année APP I & R
Novembre 2017.

(Durée 1h45)

Tous les programmes seront donnés en langage algorithmique.

Ils devront respecter scrupuleusement **TOUTES les normes de bonne programmation.**

Les exercices suivants manipulent des chaînes de caractères de type T_CHAINE défini ci-dessous.

```
-----  
CONSTANTE entier Longueur_Max <-- 100  
-----  
TYPE T_CHAINE est TABLEAU(1..Longueur_Max) de Caractères
```

1 Codage de De Vigènère

Dans cet exercice nous nous proposons d'étudier une technique de cryptage de texte par décalage.

Le principe du cryptage consiste à produire, en résultat d'un algorithme, une chaîne de caractères correspondant au cryptage d'une autre chaîne de caractères, en entrée de cet algorithme. Ce cryptage est réalisé à l'aide d'une clé qui est elle même une chaîne de caractères dans le cas du codage de De Vigènère. La chaîne de caractères en entrée peut représenter un mot à coder ou une phrase à coder (dans ce dernier cas les mots de la phrase sont séparés par un caractère ' ' c'est-à-dire un espace).

Pour le codage de De Vigènère, abordé dans cet exercice, on fait les hypothèses suivantes :

1. la chaîne de caractères à coder ne comporte que des lettres en majuscules sans accent. Aucun caractère spécial comme l'apostrophe ou le tiret ne fera partie des phrases étudiées. Seul, le caractère ' ' (espace) sera utilisé : il permet de séparer les mots d'une phrase. Le caractère ' ' (espace) est codé par lui même (et n'entre donc pas en considération).
2. les caractères alphabétiques sont indexés de 0 pour le caractère 'A' à 25 pour le caractère 'Z'. On pourra construire un tableau d'entier **code**, indexé par les caractères dont les valeurs correspondent aux différents codes des lettres. Par exemple, **code('B')** vaut 1.

Dans le codage de De Vigènère, les décalages utilisés dépendent d'une clé qui est une chaîne de caractères.

Par exemple, si la clé est "BONJOUR", les lettres du message seront décalées de :

1 lettre ('B' a pour code 1)
14 lettres ('O' a pour code 14)
13 lettres ('N' a pour code 13)
9 lettres ('J' a pour code 9)
14 lettres ('O' a pour code 14)
20 lettres ('U' a pour code 20)
17 lettres ('R' a pour code 17)

Ainsi pour coder le message "ELEVE DE PREMIERE ANNEE", avec la clé "BONJOUR" on répète la clé autant de fois que nécessaire et procède donc comme suit :

```
ELEVE DE PREMIERE ANNEE
+ BONJO UR BONJOURB ONJOU
-----
FZRES XVQFR VWYIF OAWSY
```

Le codage de la même phrase avec une autre clé 'IMPERATIVE' donne :

```
ELEVE DE PREMIERE ANNEE
+ IMPER AT IVEIMPER ATIVE
-----
MXTZV DXXMI UUTVV AGVZI
```

Questions

1. Donner le contrat d'un sous-programme de codage de De Vigénère appelé **Vigenere**.
2. Raffiner cette spécification pour obtenir un algorithme qui implante cette spécification.
3. Ecrire un programme principal qui lit une chaîne de caractères I à coder ainsi qu'une chaîne de caractères C jouant le rôle de clé et qui affiche le codage de De Vigénère de cette chaîne I . Ce programme fera appel au sous-programme **Vigenere** obtenu en réponse à la question précédente.

2 Un peu de génétique

L'ADN (AcideDesoxyriboNucléique) est composé de séquences de quatre molécules appelées bases azotées. Il s'agit de l'Adénine (A), la cytosine (C), la Guanine (G) et la Thymine (T). Un exemple simplifié d'une séquence ADN peut être **A T G G T A C G T C A A C T G A**

Les manipulations génétiques sont des opérations permettant de modifier une séquence d'ADN. On parle alors de mutation génétique. Les manipulations consistent à produire une nouvelle séquence à partir d'une séquence donnée en utilisant des règles de mutation.

Une règle de mutation $\alpha \rightarrow \beta$ est composée d'un motif source α et d'un motif cible β , elle indique que le motif source α d'une séquence est remplacé par le motif cible β dans cette même séquence. La règle $\alpha \rightarrow \Lambda$ indique que le motif α disparaît dans une séquence.

Dans cet exercice, on fera les simplifications suivantes:

1. on se limitera aux règles dans lesquelles les motifs source disparaissent, c'est-à-dire de la forme $\alpha \rightarrow \Lambda$
2. on considèrera que les règles ne sont pas ambiguës, c'est-à-dire qu'une unique règle est applicable sur un motif. Par exemple, les deux règles ci-dessous, sont non ambiguës, elles indiquent que les motifs GT et GA disparaissent.

Règle	numéro de règle
$GT \rightarrow \Lambda$	(1)
$GA \rightarrow \Lambda$	(2)

Par contre, les deux règles suivantes sont ambiguës,

Règle	numéro de règle
$GAT \rightarrow \Lambda$	(1)
$GA \rightarrow \Lambda$	(2)

En effet, sur la séquence $\dots ATGATT\dots$, toutes deux peuvent s'appliquer sur le même motif et donner $\dots ATT\dots$ ou bien $\dots ATTT\dots$.

3. Les séquences, les motifs ainsi que les règles sont représentées par des chaînes de caractères (conformes au type `T_Chaine`). Il est inutile de représenter le motif cible (membre droit) d'une règle car celui-ci est vide. Ainsi, un ensemble de règles pourra être représenté par un tableau contenant des éléments de type `T_Chaine`.

Les règles de mutation sont appliquées aussi longtemps qu'un motif source d'une règle apparaît dans la séquence considérée.

L'application de la règle (1) $GT \rightarrow \Lambda$ et (2) $GA \rightarrow \Lambda$ sur la séquence précédente donne les séquences ADN suivantes

Pas	Séquence	Numéro de règles appliquées
Séquence 0 (départ)	A T G G T A C G T C A A C T G A	
Séquence 1	A T G A C C A A C T	(1) (1) (2)
Séquence 2	A T C C A A C T	(2)
Séquence 3	A T C C A A C T	Arrêt

Questions

On souhaite écrire un algorithme qui produit une séquence d'ADN issue de la modification d'une séquence d'ADN initiale en utilisant un ensemble de règles de mutation données et sur laquelle aucune règle de mutation ne peut être appliquée.

1. Proposer un contrat pour cet algorithme
2. Raffiner la spécification obtenue jusqu'à obtenir un algorithme qui permet de produire une séquence ADN résultant de l'application des règles de mutation a été effectuée et sur laquelle aucune règle de mutation ne peut être appliquée.
3. Pourquoi cet algorithme termine-t-il ? Justifier votre réponse.

Pour cet exercice, on supposera que la procédure `DecalageGauche`, décrite ci-dessous, est disponible.

```

-- SEMANTIQUE : Les éléments de la chaîne Fseq depuis la position
--              Fdeb + Fnb_cases sont décalés à la position Fdeb.
-- PARAMETRES
--       Fseq : chaîne de caractères subissant le décalage
--       Flong : longueur de la chaîne Fseq
--       Fdeb : position à laquelle la sous-chaîne est décalée
--       Fnb_cases : nombre de cases de décalage dans le tableau
-- PRECONDITION : Fdeb+Fnb_cases<= Flong
-- POSTCONDITION : Flong = Flong'Avant - Fnb_cases
--
Procedure DecalageGauche( Fseq      : IN/OUT T_Chaine ;
                          Flong     : IN/OUT Entier;
                          Fdeb      : IN Entier ;
                          Fnb_cases : IN Entier      ) est

```