Méthodologie de la Programmation (1ère Année – Session 2)

Mai 2018.

(Durée 1h30)

— Tous les programmes demandés **seront écrits en langage algorithmique**.
— Ces programmes devront respecter scrupuleusement **TOUTES les consignes de bonne programmation définies en cours, TD et TP.**
— Pour la question A2, vous **répondrez sur la feuille numéro 2 après avoir inscrit vos nom et prénom**.

1 Questions de cours (5 points)

C1. Identifiez les différences entre la spécification et l'implantation d'un module. Argumentez votre réponse en quelques lignes.

C2. Quels sont les avantages d'un module ?

C3. Dans le cas de la déclaration et de l'appel d'un sous-programme, quel est le lien entre *appelant* et *appelé*. Justifiez votre réponse en quelques lignes.

C4. Qu'est ce qu'implique (ou signifie) de définir une pré-condition `Pre_1` sur un sous-programme `SP1` ?

C5. Quelles sont les structures de contrôle de type répétition dans notre langage algorithmique et quand les utiliser ?

2 Trouver les erreurs (6 points)

Considérons les entêtes (signatures) des deux sous-programmes suivants.

```
PROCEDURE F(X : IN OUT ENTIER)

PROCEDURE G(X : IN OUT ENTIER ; N : IN ENTIER)
```

Soient `A`, `M`, `P` et `Y` des variables de type `ENTIER`, déjà initialisées, et considérons la liste suivante des appels à ces sous-programmes

```
1. F (-Y);

2. F (Y);

3. G (P, A*P);

4. G (Y*Y, M);

5. G (Y);
```

A1. Pour chacun des appels précédents, indiquez si cet appel est correct. Les réponses doivent être justifiées.

Pour la question suivante, considérons le programme Ada `Recherche_Elt` .

NOM : **PRENOM :**

```
PROCEDURE Recherche_Elt IS

TYPE T_Tableau IS ARRAY (1 .. 10) OF INTEGER ;

FUNCTION Indice_Elt(FTab : IN T_Tableau  ; Fe:IN INTEGER) RETURN INTEGER IS

BEGIN

   WHILE I <= FTaille   AND      FTab(FIndice) /= Fe   LOOP

         FIndice := FIndice + 1 ;

   END LOOP;

   Fe := 0;

END Indice_Elt ;

-- Déclaration des variables
T : T_Tableau ;

Elt, Indice, N: INTEGER

BEGIN

   --  lecture des éléments du tableau
   --  1- Lecture/Entrée du nombre N d'éléments du tableau T

    LOOP
        Put ("Donner un entier compris entre 1 et 10");
        Get (N) ;
   EXIT WHEN N<=10 and N>=1 ;
   END LOOP ;


   --  2- Lecture/Entrée des N éléments entiers positifs  du tableau T

   PUT ("Donner les éléments du tableau ");
   FOR i IN 1..N LOOP
       GET (T(i)) ;
   END LOOP ;


   --  3- Lecture/Entrée de l'élément pour lequel
   --     on recherche l'indice dans T[1..N]

   PUT ("Donner un entier ");
   GET (Elt) ;


   -- 4- Recherche de l'indice Ind de l'élément Elt dans T[1..N]

   Indice_Elt(T, Elt, N, Indice);

   -- Ecriture/sortie de l'indice de l'élément Elt dans le tableau
   PUT ("L'indice est ");
   PUT (Indice);
END Recherche_Elt;
```

Le programme principal initialise un tableau (1 et 2), demande à l'utilisateur un entier (3), recherche et affiche l'indice du tableau où se trouve cet entier. Elle affiche 0 si l'entier ne s'y trouve pas.

Cet algorithme présente plusieurs erreurs de programmation et conception.

A2. Corrigez le programme ci-dessus pour qu'il réalise bien cette recherche.

Pour cette question, vous répondrez directement sur la feuille numéro 2 du sujet.

A3. Donnez une spécification de la fonction `Recherche_Elt`

3 Conception par raffinage (9 points)

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 $\alpha \longrightarrow \beta$ 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 $\alpha \longrightarrow \mathbf{\Lambda}$ 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 $\alpha \longrightarrow \mathbf{\Lambda}$
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
$\mathbf{GT \longrightarrow \Lambda}$	(1)
$\mathbf{GA \longrightarrow \Lambda}$	(2)

Par contre, les deux règles suivantes sont ambiguës.

Règle	numéro de règle
$\mathbf{GAT \longrightarrow \Lambda}$	(1)
$\mathbf{GA \longrightarrow \Lambda}$	(2)

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

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) $\mathbf{GT \longrightarrow \Lambda}$ et (2) $\mathbf{GA \longrightarrow \Lambda}$ 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 sous-programme 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 de ces règles de mutation ne peut être appliquée.

1. Proposez la spécification de ce sous-programme
2. Raffinez complètement cette spécification.
3. Pourquoi cet algorithme termine-t-il ? Justifiez votre réponse.