SÉRIE D'EXERCICES n° 1

EXERCICE **1** :

Soit le mot $x = ((acbc)^R.baca)^R$ (α^R désigne le reflet miroir de α)

1) Donner la chaîne de caractères à laquelle x est égal.
2) Quelle est la valeur de $|x|$, $|x|_a$, $|x|_b$ et $|x|_c$?
3) Donner un préfixe propre de x contenant au moins deux lettres 'c'.
4) Donner un suffixe propre de x contenant une seule lettre 'a'.

EXERCICE **2** :

Soit la grammaire G = ({a, b, c} , {S, A} , S , P) ; où P contient les règles suivantes :

$S \rightarrow aS \mid bA$; $A \rightarrow cA \mid \varepsilon$

1) Déterminer si les mots w_1 = abac, w_2 = aabccc, w_3 = cabbac et w_4 = ab sont générés par G.
2) Trouver le langage généré par G (qu'on note L(G)).

EXERCICE **3** :

Pour chacun des langages suivants, donner des exemples de mots contenus dans chacun des langages, et des grammaires qui engendrent chacun des langages $(L_i)_{i=1,..,5}$:

1) L_1 = { w ∈ {a, b, c}* / w commence par la lettre 'a' } ;
2) L_2 = { w ∈ {a, b, c}* / w se termine par la lettre 'a' } ;
3) L_3 = { w ∈ {a, b, c}* / w contient au moins une occurrence de la lettre 'a' } ;
4) L_4 = { w ∈ {a, b, c}* / w contient au moins deux occurrences la lettre 'a' } ;
5) L_5 = { w ∈ {a, b, c}* / w contient au moins deux occurrences consécutives de la lettre 'a' }.

EXERCICE **4** :

Soient les grammaires G_i = ({a, b, c} , {S, A, B, R, T} , S , P_i), (i=1,..,6) ; où les P_i sont :

1) P_1 : $S \rightarrow aA \mid bB$; $A \rightarrow a \mid ab$; $B \rightarrow b \mid cb$
2) P_2 : $S \rightarrow bA$; $A \rightarrow aA \mid \varepsilon$
3) P_3 : $S \rightarrow aSc \mid A$
   $A \rightarrow bA \mid b$
4) P_4 : $S \rightarrow aSbS \mid \varepsilon$
5) P_5 : $S \rightarrow aRbc \mid abc$
   $R \rightarrow aRTb \mid aTb$; $Tb \rightarrow bT$; $Tc \rightarrow cc$
6) P_6 : $S \rightarrow aAb \mid \varepsilon$
   $A \rightarrow aSb$
   $Ab \rightarrow \varepsilon$

I) Pour chacune des grammaires G_i (i=1,..,6) ; donner le type de celle-ci, puis trouver le langage engendré par chacune d'elles.

II) Vérifier que G_2 n'est pas de type 1 ; mais que $L(G_2)$ est de type 1.

III) Montrer que $L(G_6)$ est de type 2 en trouvant une grammaire de type 2 qui l'engendre.

EXERCICE **5** :

Pour chacun des langages suivants, donner une grammaire qui l'engendre :

a) $L_1 = \{ 0^{2n} / \ n \geq 0 \}$

b) $L_2 = \{ 0^n 1^n / \ n \geq 0 \}$

c) $L_3 = \{ a^n b^{2n} / \ n \geq 0 \}$

d) $L_4 = \{ a^n b^m c^{n-m} / \ n \geq m \geq 0 \}$

e) $L_5 = \{$ palindromes de $\{a, b\}^* \}$

f) $L_6 = \{ a^m b^n a^n b^m / \ n \geq 1, m \geq 1 \}$

g) $L_7 = \{ w \in \{a, b\}^* / \ |w| \equiv 0[3] \}$

h) $L_8 = \{ 0^i 1^j / \ i \geq j \geq 0 \}$

i) $L_9 = \{ 0^i 1^j / \ i \neq j, i \geq 0, j \geq 0 \}$

j) $L_{10} = \{ a^{2^n} / \ n \geq 0 \}$

EXERCICE **6** :

1) Soit L un langage de type $i \in \{0, 1, 2, 3\}$. Est-il possible qu'un langage $L' \subset L$ ne soit pas de type i ?
indication : on sait que le langage $\{ 0^n 1^n / \ n \geq 0 \}$ n'est pas régulier.

2) Montrer que toute grammaire régulière est aussi à contexte libre. Qu'en est-il de la réciproque ?

EXERCICE **7** :

Soit le langage L défini comme suit : $L = \{ a^{2.n}.b.c^{2.m+1} / \ n, m \geq 0 \}$.

1) Montrer que L est de type 3 en trouvant une grammaire de type 3 qui l'engendre.

2) Trouver une grammaire de type 2, et qui n'est pas de type 3, qui engendre L.

EXERCICE **8** :

Soit le langage L défini comme suit :

L = ensemble de tous les mots de $\{0, 1\}^*$ qui contiennent un nombre pair de '1'.

Mêmes questions, pour L, que l'exercice 7.

EXERCICE **9** :

Soit la grammaire G dont les règles de production sont :

$S \rightarrow AB \mid \varepsilon$

$A \rightarrow aAb \mid \varepsilon$

$bB \rightarrow Bbb$

$B \rightarrow \varepsilon$

1) Déterminer L(G).

2) Construire une grammaire de type 2 équivalente à G.

EXERCICE **10** :

Soit la grammaire G dont les règles de production sont :

$S \rightarrow RD$

$R \rightarrow aRb \mid A$

$Ab \rightarrow bbA$

$AD \rightarrow \varepsilon$

Mêmes questions, pour G, que l'exercice 9.

EXERCICE **11** :

Soit l'alphabet terminal $\pi = \{ p, \neg, \wedge, \vee, \Rightarrow, (,) \}$.

1) Soit L le langage des expressions de logique propositionnelle construites sur l'alphabet π.
   Trouver une grammaire, de type 2, pour L.
2) Soit L' le langage des expressions de logique propositionnelle défini sur π tel que l'on ne peut pas avoir des parenthèses imbriquées directement l'une dans l'autre. À titre d'exemple, $(p \vee (p \wedge p))$ ou (p) sont dans L' mais $(p \vee ((p \wedge p)))$ ou (((p))) ne le sont pas.
   Trouver une grammaire, de type 2, pour L'.

EXERCICE **12** :

Ecrire une grammaire pour générer les identificateurs d'un langage comme Pascal. On considérera qu'un identificateur est valide s'il commence par une lettre alphabétique (majuscule ou minuscule) qui peut éventuellement être suivie d'une ou plusieurs lettres alphabétiques et/ou chiffres.

Pour ce qui est des non terminaux de cette grammaire on pourra utiliser par exemple <Id1>, <Id2>, <Id3>, <Lettre>, <Chiffre>, …

EXERCICE **13** :

On définit une fonction δ sur : $\{a, b\}^* \rightarrow \mathbb{N}$ comme suit :

$\delta(\varepsilon) = 0$; $\delta(a) = -1$; $\delta(b) = +1$; $\delta(u.v) = \delta(u) + \delta(v)$ pour tous $u, v \in \{a, b\}^*$.

On appelle mot de Lukasiewicz un mot $u = u_1.u_2.\ldots.u_n$, où $u_i \in \{a, b\}$ pour i=1,..,n et tel que :

$$\sum_{i=1}^{n} \delta(u_i) = -1 \quad \text{et} \quad \sum_{i=1}^{k} \delta(u_i) \geq 0 \quad \text{pour k = 1,..,n-1.}$$

1) Donner tous les mots de Lukasiewicz de longueur 1, 2 et 3 ; puis tous les mots de longueur paire.
2) Montrer que si u et v sont deux mots de Lukasiewicz alors : b.u.v est un mot de Lukasiewicz.
3) Réciproquement, montrer que tout mot de Lukasiewicz de longueur supérieure ou égale à 3 admet une décomposition de la forme b.u.v, où u et v sont des mots de Lukasiewicz.
4) En déduire une grammaire à contexte libre engendrant l'ensemble de tous les mots de Lukasiewicz. On appellera L ce langage.
5) Écrire une fonction et/ou une procédure en Pascal qui reçoit en entrée une chaîne de caractères et renvoie en sortie un booléen = vrai si et seulement si la chaîne entrée $\in$ L.
   Réaliser cela de deux manières :
   5-a) en utilisant les propriétés des mots de L données au début de l'exercice ;
   5-b) en utilisant la grammaire trouvée en 4).

----------Fin Série 1----------

CORRIGÉ ABRÉGÉ DE LA SÉRIE D'EXERCICES n° 1 de ThL

par : **M.S. Habet, Y. Yesli, C. Cherifi**

EXERCICE **1** :

1) x = acabacbc
2) $|x| = 8$, $|x|_a=3$, $|x|_b=2$, et $|x|_c=3$
3) acabac
4) acbc

EXERCICE **2** :

1) Les mot w_1 et w_3 ne sont pas générés par G ;
les mots w_2 et w_4 sont générés par G : $S \vdash aS \vdash aaS \vdash aabA \vdash aabcA \vdash aabccA \vdash aabcccA \vdash w_2$
et pour w_4 : $S \vdash aS \vdash abA \vdash ab = w_4$.
2) Soit $L = \{ a^i b c^j / i, j \geq 0 \}$. Montrons que L(G)=L en prouvant la double inclusion :
   - $L(G) \subseteq L$: soit w un mot de L(G), donc w est généré à partir de S en appliquant n fois les règles de production de G. Montrons par récurrence sur n que $w \in L$:
     - si n=2 alors on a : $S \vdash bA \vdash b$; $w=b \in L$. Supposons que la propriété reste vraie jusqu'au rang n=k.
     - pour n=k+1, on a deux cas :
       -- la première règle appliquée est $S \rightarrow aS$, puis k règles pour avoir un mot a.u. Puisque u est généré à partir de S avec application de k règles de G, et d'après l'hypothèse de récurrence, u est dans L, donc il s'écrit comme $u = a^i b c^j$ et ainsi le mot $a.u = a^{i+1} b c^j \in L$.
       -- la première règle appliquée est $S \rightarrow bA$, puis à partir de A, on obtient c^j ($j \geq 0$), et on aura donc généré le mot $b.c^j$ qui $\in L$ (c'est : $a^i.b.c^j$ avec i=0).
   - $L \subseteq L(G)$: Soit $w \in L$. Donc w s'écrit comme $w= a^n b c^m$. w peut être dérivé de S en appliquant n fois la règle $S \rightarrow aS$ puis une fois la règle $S \rightarrow bA$, puis encore m fois la règle $A \rightarrow cA$ et enfin une fois la règle $A \rightarrow \varepsilon$. Donc $w \in L(G)$.

EXERCICE **3** :

1) Exemples de mots de L_1 : a, ab, accb, …
Une grammaire pour L_1 : $G_1 = (\{a, b, c\}, \{S, A\}, S, \{ S \rightarrow aA ; A \rightarrow aA \mid bA \mid cA \mid \varepsilon \})$

2) Exemples de mots de L_2 : a, ba, cabca, …
Une grammaire pour L_2 : $G_2 = (\{a, b, c\}, \{S\}, S, \{ S \rightarrow aS \mid bS \mid cS \mid a \})$

3) Exemples de mots de L_3 : a, bba, acaacb, …
Une grammaire pour L_3 : $G_3 = (\{a, b, c\}, \{S, A\}, S, \{ S \rightarrow aS \mid bS \mid cS \mid aA ; A \rightarrow aA \mid bA \mid cA \mid \varepsilon \})$

4) Exemples de mots de L_4 : aa, aba, acacab, …
Une grammaire pour L_4 : $G_4 = (\{a, b, c\}, \{S, A, B\}, S, P_4)$
P_4 : $\{ S \rightarrow aS \mid bS \mid cS \mid aA ; A \rightarrow aA \mid bA \mid cA \mid aB ; B \rightarrow aB \mid bB \mid cB \mid \varepsilon \})$

5) Exemples de mots de L_5 : aa, baab, accaab, …
Une grammaire pour L_5 : $G_5 = (\{a, b, c\}, \{S, A\}, S, \{ S \rightarrow aS \mid bS \mid cS \mid aaA ; A \rightarrow aA \mid bA \mid cA \mid \varepsilon \})$

EXERCICE **4** :

I) Nous donnons ici les types des G_i, (i=1,..,6), ainsi que les langages engendrés par les grammaires G_i (i=1,..,6). (Pour que la réponse soit complète, il faut le prouver comme c'est fait dans l'exercice 2).

1) Type de $G_1 = 3$. $L(G_1) = \{ aa, aab, bb, bcb \}$.
2) Type de $G_2 = 3$. $L(G_2) = \{ b.a^n / n \geq 0 \}$.
3) Type de $G_3 = 2$. $L(G_3) = \{ a^n b^m c^n / n \geq 0, m \geq 1 \}$.
4) Type de $G_4 = 2$. $L(G_4) = \{ w \in \{a, b\}^* / |w|_a = |w|_b$ et $\forall$ u préfixe de w, $|u|_a \geq |u|_b \}$.
5) Type de $G_5 = 1$. $L(G_5) = \{ a^n b^n c^n / n \geq 1 \}$.
6) Type de $G_6 = 0$. $L(G_6) = \{ a^n b^{2.[n/2]} / n \geq 0 \}$; ([x] est la partie entière de x)
On peut aussi écrire $L(G_6)$ comme $\{ a^{2k+1} b^{2k} / k \geq 0 \} \cup \{ a^{2k} b^{2k} / k \geq 0 \}$.

II) G_2 n'est pas de type 1 car elle contient la règle : $A \rightarrow \varepsilon$; or dans les grammaires de type 1, le seul symbole qui peut produire la chaîne vide est S.
Cependant, on peut écrire une grammaire de type 1 équivalente à G_2 :
G_2' a pour règles de production : $S \rightarrow Sa \mid b$; ce qui veut dire que $L(G_2)$ est de type 1.

III) Une grammaire de type 2 équivalente à G_6 :
G_6' a pour règles de production : $S \rightarrow aaSbb \mid a \mid \varepsilon$

EXERCICE **5** :

a) pour L_1 : il est engendré par $G_1 = (\{0\}, \{S\}, S, P_1)$, où $P_1 : S \rightarrow 00S \mid \varepsilon$
b) pour L_2 : il est engendré par $G_2 = (\{0, 1\}, \{S\}, S, P_2)$, où $P_2 : S \rightarrow 0S1 \mid \varepsilon$
c) pour L_3 : il est engendré par $G_3 = (\{a, b\}, \{S\}, S, P_3)$, où $P_3 : S \rightarrow aSbb \mid \varepsilon$
d) pour L_4 : il est engendré par $G_4 = (\{a, b\}, \{S, A\}, S, P_4)$,
où $P_4 : S \rightarrow aSc \mid A$; $A \rightarrow aAb \mid \varepsilon$
e) pour L_5 : il est engendré par $G_5 = (\{a, b\}, \{S\}, S, P_5)$,
où $P_5 : S \rightarrow aSa \mid bSb \mid a \mid b \mid \varepsilon$
f) pour L_6 : il est engendré par $G_6 = (\{a, b\}, \{S, A\}, S, P_6)$,
où $P_6 : S \rightarrow aSb \mid aAb$;
$A \rightarrow bAa \mid ba$
g) pour L_7 : il est engendré par $G_7 = (\{a, b\}, \{S, A\}, S, P_7)$,
où $P_7 : S \rightarrow AAAS \mid \varepsilon$;
$A \rightarrow a \mid b$
h) pour L_8 : il est engendré par $G_8 = (\{0, 1\}, \{S\}, S, P_8)$,
où $P_8 : S \rightarrow 0S1 \mid 0S \mid \varepsilon$

i) $L_9 = \{ 0^i 1^j / \ i > j \} \cup \{ 0^i 1^j / \ i < j \}$; L_9 est engendré par $G_9 = (\{0, 1\}, \{S, S_0, S_1\}, S, P_9)$,
où P_9 : $S \rightarrow S_0 \mid S_1$;
$S_0 \rightarrow 0S_01 \mid 0S_0 \mid 0$;
$S_1 \rightarrow 0S_11 \mid S_11 \mid 1$

j) L_{10} : il est engendré par $G_{10} = (\{0, 1\}, \{S, A, B, C, D\}, S, P_{10})$,
où P_{10} : $S \rightarrow BCD$
$C \rightarrow AC \mid a$
$Aa \rightarrow aaA$
$AD \rightarrow D$
$Ba \rightarrow aB$
$BD \rightarrow \varepsilon$

EXERCICE **6** :

1) Soient les langage $L = \{0, 1\}^*$ et $L' = \{ 0^n1^n / \ n \geq 0 \}$. L est de type 3 (vérifier le !) ; mais L', qui est inclus dans L, n'est pas de type 3 (il est de type 2).
2) On considérera le cas d'une grammaire régulière à droite (l'autre cas à gauche est aussi vérifié).
Toute grammaire régulière à droite respecte la condition que toute règle est de la forme $A \rightarrow wB \mid w$; où A est un non terminal, w un mot quelconque de terminaux (éventuellement vide) et B non terminal.
Ces contraintes respectent aussi les conditions du type 2 ; à savoir des règles de la forme $A \rightarrow w$; où A est un non terminal et w est, dans ce cas, une suite quelconque (éventuellement vide) de terminaux et/ou de non terminaux. Donc toute grammaire régulière est à contexte libre.
Mais la réciproque est fausse ; soit la grammaire à contexte libre (type 2) :
$G_2 = (\{a, b\}, \{S\}, S, \{ S \rightarrow aSb \mid \varepsilon \})$. Elle vérifie les conditions du type 2 mais pas celles du type 3.

EXERCICE **7** :

1) L peut être généré par la grammaire, de type 3, $G = (\{a, b, c\}, \{S, C\}, S, P)$
où P : $S \rightarrow aaS \mid bcC$
$C \rightarrow ccC \mid \varepsilon$
2) Une autre grammaire de type 2, et qui n'est pas de type 3, qui engendre L :
$G' = (\{a, b, c\}, \{S, A, C\}, S, P')$
où P' : $S \rightarrow AbcC$
$A \rightarrow aaA \mid \varepsilon$
$C \rightarrow ccC \mid \varepsilon$

EXERCICE **8** :

1) L peut être généré par la grammaire, de type 3, $G = (\{0, 1\}, \{S, A\}, S, P)$
où P : $S \rightarrow 0S \mid 1A \mid \varepsilon$
$A \rightarrow 0A \mid 1S$
2) Une autre grammaire de type 2, et qui n'est pas de type 3, qui engendre L :
$G' = (\{0, 1\}, \{S\}, S, P')$
où P' : $S \rightarrow 0S \mid S1S1S \mid \varepsilon$

EXERCICE **9** :

1) $L(G) = \{ a^n b^m / n \leq m \leq 2*n \}$

2) Grammaire à contexte libre équivalente à G : G' = ({a, b}, {S, B}, S, P')

P' : $S \rightarrow aSbB \mid \varepsilon$

$B \rightarrow b \mid \varepsilon$

EXERCICE **10** :

1) $L(G) = \{ a^n b^{2n} / n \geq 0 \}$;

2) Grammaire de type 2 équivalente à G : G' = ({a, b}, {S}, S, P')

où P' : $S \rightarrow aSbb \mid \varepsilon$

EXERCICE **11** :

1) Une grammaire de type 2 pour L pourrait être G = (π, N, S, P) ; où N = {S}

et P : $S \rightarrow p \mid \neg S \mid S \wedge S \mid S \vee S \mid S \Rightarrow S \mid (S)$

2) Une grammaire de type 2 pour L' pourrait être G' = (π, N', S, P') ; où N' = {S, A}

et P' : $S \rightarrow p \mid \neg S \mid S \wedge S \mid S \vee S \mid S \Rightarrow S \mid (A)$

$A \rightarrow p \mid \neg S \mid S \wedge S \mid S \vee S \mid S \Rightarrow S$

EXERCICE **12** :

Pour générer ces identificateurs on utilisera la grammaire G = (π, N, <Id1>, P) ;

où π = {A..Z, a..z, 0..9} ; N = {<Id1>, <Id2>, <Id3>, <Lettre>, <Chiffre>}

et P : <Id1> → <Lettre> <Id2>

<Id2> → <Id3> <Id2> | ε

<Id3> → <Lettre> | <Chiffre>

<Lettre> → A | B | .. | Z | a | b | .. | z

<Chiffre> → 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

EXERCICE **13** :

1) Soit un u mot de Lukasiewicz. On peut Remarquer que si $\sum_{i=1}^{n} \delta(u_i) = -1$ alors $\sum_{i=1}^{n-1} \delta(u_i) = 0$.

Donc $\delta(u_n) = -1$, qui veut dire que u_n = 'a' ou bien u se termine par un 'a'.

- Il y a un seul mot de Lukasiewicz de longueur 1 ; c'est 'a'.
- Il n'y a pas de mot de longueur 2. S'il y en avait, cela signifierait que $\delta(u_1) = 0$, ce qui est impossible car u_1 n'est pas égal à ε.
- Il y a un seul mot de longueur 3, c'est 'baa' ; cela se déduit des équations/inéquations :

$\delta(u_1) + \delta(u_2) = 0$

$\delta(u_1) \geq 0$

$\delta(u_3) = -1$

desquelles on déduit que $\delta(u_1) = +1$ et $\delta(u_2) = -1$; donc u_1 = 'b', u_2 = 'a' et u_3 = 'a'.

- Il n'y a pas de mot de longueur paire : d'après la formule $\sum_{i=1}^{n-1} \delta(u_i) = 0$; qui signifie qu'il y a autant de 'b' que de 'a' dans $u_1.u_2.....u_{n-1}$. Par conséquent n-1 est pair, et ainsi n est impair.

2) Soient u et v deux mots de Lukasiewicz de longueurs respectives n et m. Chacun de ces mots vérifie les équations données dans l'exercice. Le mot b.u.v qu'on note w est construit comme :

w1='b' ; $w_i = u_{i-1}$ pour i=2,..,n+1 et $w_i = v_{i-(n+1)}$ pour i=n+2,..,n+m+1.
w vérifie aussi les équations de l'exercice :

-- $\sum_{i=1}^{n+m+1} \delta(w_i) = +1 + \sum_{i=1}^{n} \delta(u_i) + \sum_{i=1}^{m} \delta(v_i) = +1-1-1 = -1$

-- $\sum_{i=1}^{k} \delta(w_i) = +1 + \sum_{i=1}^{k-1} \delta(u_i) \geq 0$ pour k = 1,..,n+1 et aussi :

$\sum_{i=1}^{k} \delta(w_i) = +1 + (-1) + \sum_{i=1}^{k-n-1} \delta(v_i) \geq 0$ pour k = n+2,..,n+m.

D'où le résultat.

3) Soit w un mot de Lukasiewicz de longueur n.

-- Cas initial n=3 :

On a déjà vu en 1) que le seul mot de Lukasiewicz de longueur 3 est 'baa'

-- Cas général :

On sait que $\delta(w_1) = +1$, d'après la 2ième propriété ; donc $w_1 = b$.

Du même procédé qu'en 1), on déduit que $w_n = a$.

De la deuxième propriété appliquée comme en 1) on déduit qu'il existe j tel que :

$2 \leq j \leq n-1$ et $\sum_{i=1}^{j} \delta(u_i) = 0$. Ans ce cas, il suffira de prendre $u = w_2 \ldots w_j$ et $v = w_{j+1} \ldots w_n$.

Si u = 'a' alors u est un mot de Lukasiewicz, et v aussi (vérifier le).

Si v = 'a' alors v est un mot de Lukasiewicz, et u aussi (vérifier le).

Sinon $3 \leq j \leq n-2$, et dans ce cas on peut déduire ce qui suit :

$\sum_{i=1}^{j-1} \delta(u_i) = \sum_{i=2}^{j} \delta(w_i) = -1$

$\sum_{i=1}^{k} \delta(u_i) = \sum_{i=2}^{k+1} \delta(wi) \geq 0$ pour k = 1,..,(j-2).

Ainsi u est mot de Lukasiewicz. Et aussi :

$\sum_{i=1}^{n-j} \delta(v_i) = \sum_{i=j+1}^{n} \delta(w_i) = -1$

$\sum_{i=1}^{k} \delta(v_i) = \sum_{i=j+1}^{k+j} \delta(wi) \geq 0$ pour k = 1,..,(n-j-1).

Ce qui veut dire que v est un mot de Lukasiewicz.

4) Une grammaire pour L : G = ({a, b}, {S}, S, { $S \rightarrow bSS \mid a$ }).

5) Les programmes Pascal de reconnaissance des mots de Lukasiewicz.

5-a) Le programme contiendra une fonction Verifa qui implémente les équations/inéquations données dans l'énoncé de l'exercice.

On utilisera les variables :

c de type string : c'est la chaîne à vérifier
luk de type boolean : contiendera le résultat à retourner
i, som de type integer : i pour la chaine et som pour les sommes partielles des $\delta(w_i)$

```
Program Exo13a;
uses wincrt;
var c : string;
    luk : boolean;
    i,som : integer;

Function Verifa : boolean;
begin
  if c='' then luk := false
  else
   if c='a' then luk := true
   else begin
     som := 0; i:=1; luk := true;
     while (i<length(c)) and (som>=0) and luk do
     begin
       if c[i]='a' then
         som := som-1
       else
         if c[i]='b' then
           som := som+1
         else
           luk := false;
       i := i+1
     end;
     luk := luk and (som=0) and (c[i]='a');
   end;
   Verifa := luk
end; (* fin de Verifa *)

Begin
  writeln('donner un mot : ');
  readln(c);
  if Verifa then
    writeln ('c''est un mot de Lukasiewicz')
  else
    writeln('ce n''est pas un mot de Lukasiewicz');
End.
```

5-b) Avant d'écrire le programme de vérification, on modifie un peu la grammaire de 4)
En ajoutant un nouveau non terminal S', qui sera le nouvel axiome, et la règle S' → S#
On associera, dans le programme, à chaque non terminal une procédure Pascal :
à S' on écrit la procédure Verifb ;
à S on écrit la procédure (récursive) S.
Chaque procédure sera codée suivant les règles de production du non terminal associé.
La chaîne à vérifier sera stockée dans la variable c de type string ; elle sera parcourue avec l'indice i : variable de type integer.
Le résultat est retourné dans la variable luk de type boolean.
On obtient ainsi ce qui suit :

```
Program Exo13b;
uses wincrt;
var c : string;
    luk : boolean;
    n,i : integer;

Procedure Verifb;
 procedure S;
 begin
   if (i<=n) and luk then begin
    if c[i]='a' then i:= i+1
    else
      if c[i]='b' then begin
        i:=i+1;
        S;
        if i>n then
          luk := false
        else
          S;
      end
      else luk := false;
   end
 end; (* fin de S*)
begin
  i:=1; luk := true;
  c := c + '#';
  S;
  luk := luk and (c[i]='#');
end; (* fin de Verifb *)

Begin
  writeln('donner un mot : ');
  readln(c);
  if c='' then luk := false
  else
    if c='a' then luk := true
    else begin
      n := length(c);
      if (n>=3) then
        Verifb
      else
        luk := false;
    end;
  if luk then
    writeln ('c''est un mot de Lukasiewicz')
  else
    writeln('ce n''est pas un mot de Lukasiewicz');
End.
```

---------- Fin du corrigé de la série n° 1 de ThL ----------