

Site : Luminy St-Charles St-Jérôme Cht-Gombert Aix-Montperrin Aubagne-SATIS
 Sujet de : 1^{er} semestre 2^{ème} semestre Session 2 Durée de l'épreuve : 2h
 Examen de : L3 Nom du diplôme : Licence d'Informatique
 Code du module : ENSIN6U1 Libellé du module : Compilation
 Calculatrices autorisées : NON Documents autorisés : NON

1 Chemins

On considère un point dans le plan, identifié par ses coordonnées cartésiennes (x, y) . Ce point est initialement situé aux coordonnées $(0, 0)$ et peut effectuer quatre mouvements élémentaires d'une unité, vers le haut, le bas, la gauche et la droite. À chacun de ces mouvements correspond une lettre : H, B, G et D .

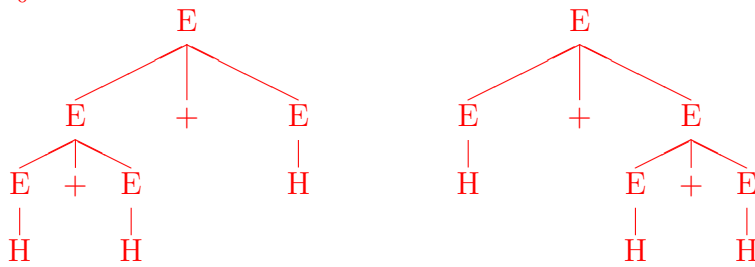
Un chemin est une séquence de mouvements représentée sous la forme d'une expression arithmétique. L'expression $B + G$ correspond à une séquence d'un mouvement élémentaire vers le bas suivi d'un mouvement élémentaire vers la gauche. La répétition de mouvements peut être indiquée grâce à l'opérateur \times . L'expression $3 \times G$ correspond à une séquence de trois mouvements élémentaires vers la gauche. L'opérateur \times est plus prioritaire que l'opérateur $+$ (l'expression $3 \times G + B$ correspond à $(3 \times G) + B$). L'usage de parenthèses permet de répéter un chemin complexe : $3 \times (H + 7 \times (B + 3 \times G))$.

Soit la grammaire G_0 suivante, d'axiome E , permettant de générer de telles expressions :

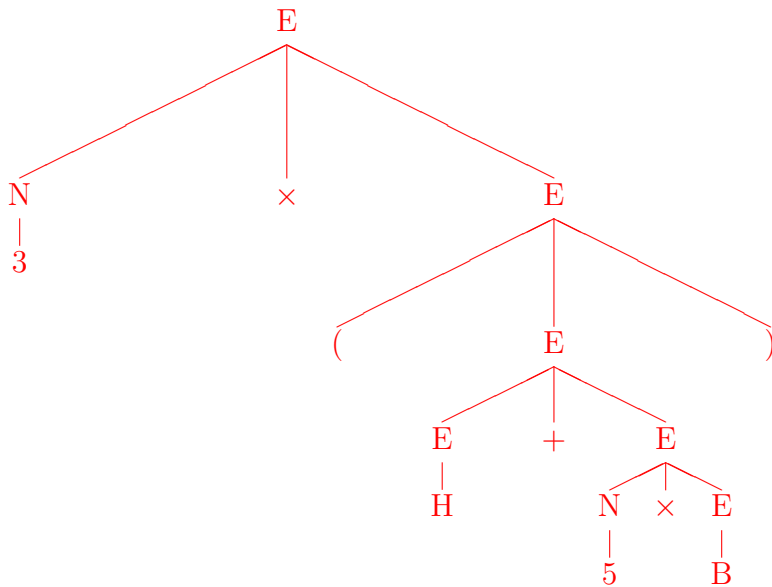
$$\begin{array}{ll} E \rightarrow E + E & E \rightarrow D \\ E \rightarrow N \times E & N \rightarrow NC \\ E \rightarrow (E) & N \rightarrow C \\ E \rightarrow H & C \rightarrow 0 \\ E \rightarrow B & \dots \\ E \rightarrow G & C \rightarrow 9 \end{array}$$

Question 1.1 Montrer que la grammaire G_0 est ambiguë

G_0 associe deux arbres de dérivation au mot $H + H + H$:



Question 1.2 Dessiner un arbre de dérivation correspondant à l'expression $3 \times (H + 5 \times B)$



Question 1.3 Ecrire une grammaire G_1 non ambiguë telle que $L(G_1) = L(G_0)$

$$\begin{array}{ll}
 E \rightarrow E + E' & E' \rightarrow D \\
 E \rightarrow E' & E' \rightarrow NC \\
 E' \rightarrow N \times E' & N \rightarrow C \\
 E' \rightarrow (E) & C \rightarrow 0 \\
 E' \rightarrow H & \dots \\
 E' \rightarrow B & C \rightarrow 9 \\
 E' \rightarrow G &
 \end{array}$$

Question 1.4 Ecrire une grammaire G_2 non réursive à gauche telle que $L(G_2) = L(G_0)$

$$\begin{array}{ll}
 E \rightarrow E'E'' & E' \rightarrow D \\
 E'' \rightarrow +E'E'' & E' \rightarrow NC \\
 E'' \rightarrow \varepsilon & N \rightarrow C \\
 E' \rightarrow N \times E' & N \rightarrow C \\
 E' \rightarrow (E) & C \rightarrow 0 \\
 E' \rightarrow H & \dots \\
 E' \rightarrow B & C \rightarrow 9 \\
 E' \rightarrow G &
 \end{array}$$

Question 1.5 Ecrire une grammaire G_3 factorisée à gauche telle que $L(G_3) = L(G_0)$

$$\begin{array}{ll}
 1 \ E \rightarrow E'E'' & 9 \ E' \rightarrow D \\
 2 \ E'' \rightarrow +E'E'' & 10 \ N \rightarrow CN' \\
 3 \ E'' \rightarrow \varepsilon & 11 \ N' \rightarrow N \\
 4 \ E' \rightarrow N \times E' & 12 \ N' \rightarrow \varepsilon \\
 5 \ E' \rightarrow (E) & 13 \ C \rightarrow 0 \\
 6 \ E' \rightarrow H & \dots \\
 7 \ E' \rightarrow B & 22 \ C \rightarrow 9 \\
 8 \ E' \rightarrow G &
 \end{array}$$

Question 1.6 Calculer les premiers et suivants des symboles de G_3

	<i>PREMIERS</i>	<i>SUIVANTS</i>
<i>E</i>	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, (, <i>H</i> , <i>B</i> , <i>G</i> , <i>D</i>	⊥,)
<i>E'</i>	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, (, <i>H</i> , <i>B</i> , <i>G</i> , <i>D</i>	+, ⊥,)
<i>E''</i>	+, ε	⊥,)
<i>N</i>	0, 1, 2, 3, 4, 5, 6, 7, 8, 9	×
<i>N'</i>	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, ε	×
<i>C</i>	0, 1, 2, 3, 4, 5, 6, 7, 8, 9	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, ×

Question 1.7 Dessiner la table *LL*(1) de G_3 . G_3 est-elle *LL*(1) ? Si ce n'est pas le cas, modifiez là afin qu'elle le devienne.

	+	×	()	<i>G</i>	<i>H</i>	<i>B</i>	<i>D</i>	0	...	9	⊥
<i>E</i>					1	1	1	1	1	1	1	
<i>E'</i>			5		8	6	7	9	4	4	4	
<i>E''</i>	2			3								3
<i>N</i>									10	10	10	
<i>N'</i>		12							11	11	11	
<i>C</i>									13	...	22	

Question 1.8 Simuler l'analyse de l'expression $3 \times (H + 5 \times B)$ à l'aide de la table *LL*.

<i>PILE</i>	<i>MOT</i>	<i>PRODUCTION</i>
<i>E</i>	$3 \times (H + 5 \times B) \perp$	1
<i>E''E'</i>	$3 \times (H + 5 \times B) \perp$	4
<i>E''E' \times N</i>	$3 \times (H + 5 \times B) \perp$	16
<i>E''E' \times 3</i>	$3 \times (H + 5 \times B) \perp$	
<i>E''E' \times</i>	$\times (H + 5 \times B) \perp$	
<i>E''E'</i>	$(H + 5 \times B) \perp$	5
<i>E''E</i> ($(H + 5 \times B) \perp$	
<i>E''E</i>) <i>E</i>	$H + 5 \times B) \perp$	1
<i>E''E</i>) <i>E''E'</i>	$H + 5 \times B) \perp$	6
<i>E''E</i>) <i>E''H</i>	$H + 5 \times B) \perp$	
<i>E''E</i>) <i>E''</i>	$+5 \times B) \perp$	2
<i>E''E</i>) <i>E''E' +</i>	$+5 \times B) \perp$	
<i>E''E</i>) <i>E''E'</i>	$5 \times B) \perp$	4
<i>E''E</i>) <i>E''E' \times N</i>	$5 \times B) \perp$	10
<i>E''E</i>) <i>E''E' \times N'C</i>	$5 \times B) \perp$	18
<i>E''E</i>) <i>E''E' \times N'5</i>	$5 \times B) \perp$	
<i>E''E</i>) <i>E''E' \times N'</i>	$\times B) \perp$	12
<i>E''E</i>) <i>E''E' \times</i>	$\times B) \perp$	
<i>E''E</i>) <i>E''E'</i>	$B) \perp$	7
<i>E''E</i>) <i>E''B</i>	$B) \perp$	
<i>E''E</i>) <i>E''</i>	$) \perp$	3
<i>E''E</i>)	$) \perp$	
<i>E''E</i>	\perp	3
	\perp	

On repart maintenant de la grammaire G_0 . Dans chacune des questions suivantes on cherche à répondre à une question concernant une expression générée par la grammaire. Pour chacune, il vous est demandé de définir un ou plusieurs attributs ainsi que des actions sémantiques associées aux règles de G_0 permettant de calculer la valeur des attributs. Tous les attributs sont à valeurs entières. Pour chaque attribut défini, vous indiquerez : ce qu'il représente et s'il est hérité ou synthétisé

Question 1.9 Combien de mouvements élémentaires comporte un chemin ? L'expression $3 \times (4 \times G + D)$ par exemple, comporte 15 mouvements élémentaires.

on définit l'attribut synthétisé M qui compte le nombre de mouvements élémentaires d'un chemin

$$\begin{array}{l|l}
 E \rightarrow E + E & E_0.M = E_1.M + E_2.M \\
 E \rightarrow N \times E & E_0.M = N.M \times E_1.M \\
 E \rightarrow (E) & E_0.M = E_1.M \\
 E \rightarrow H & E_0.M = 1 \\
 E \rightarrow B & E_0.M = 1 \\
 E \rightarrow G & E_0.M = 1 \\
 E \rightarrow D & E_0.M = 1 \\
 N \rightarrow NC & N_0.M = 10 \times N_1.M + C.M \\
 N \rightarrow C & N_0.M = C.M \\
 C \rightarrow 0 & C_0.M = 0 \\
 C \rightarrow 1 & C_0.M = 1 \\
 \dots & \\
 C \rightarrow 9 & | C_0.M = 9
 \end{array}$$

Question 1.10 Quelles sont les coordonnées du point à l'issue du chemin ? L'expression $G + H + D + B$, par exemple, aboutit au point de coordonnées $(0, 0)$.

on définit les deux attribut synthétisés x et y

$$\begin{array}{l|l}
 E \rightarrow E + E & E_0.x = E_1.x + E_2.x & E_0.y = E_1.y + E_2.y \\
 E \rightarrow N \times E & E_0.x = N.M \times E_1.x & E_0.y = N.M \times E_1.y \\
 E \rightarrow (E) & E_0.x = E_1.x & E_0.y = E_1.y \\
 E \rightarrow H & E_0.x = 1 & E_0.y = 0 \\
 E \rightarrow B & E_0.x = -1 & E_0.y = 0 \\
 E \rightarrow G & E_0.x = 0 & E_0.y = -1 \\
 E \rightarrow D & E_0.x = 0 & E_0.y = 1 \\
 N \rightarrow NC & & \\
 N \rightarrow C & & \\
 C \rightarrow 0 & & \\
 C \rightarrow 1 & & \\
 \dots & & \\
 C \rightarrow 9 & | &
 \end{array}$$

Question 1.11 Quelle est l'abscisse maximale de la trajectoire suivie par le point ? Pour l'expression $D + H + G + B$, par exemple, l'abscisse maximale vaut 1.

on définit l'attribut hérité x et l'attribut synthétisé m . x représente l'abscisse courant et m l'abscisse maximal. On ajoute la règle $S \rightarrow E$ afin d'initialiser l'attribut x .

$$\begin{array}{l|l}
 S \rightarrow E & E_1.x = 0 \\
 E \rightarrow E + E & E_1.x = E_0.x & E_2.x = E_1.m & E_0.m = \max(E_1.m, E_2.m) \\
 E \rightarrow N \times E & E_1.x = E_0.x & & E_0.m = N.M \times E_1.m \\
 E \rightarrow (E) & E_1.x = E_0.x & & \\
 E \rightarrow H & & & E_0.m = E_0.x + 1 \\
 E \rightarrow B & & & E_0.m = E_0.x - 1 \\
 E \rightarrow G & & & E_0.m = E_0.x \\
 E \rightarrow D & & & E_0.m = E_0.x \\
 N \rightarrow NC & & & \\
 N \rightarrow C & & & \\
 C \rightarrow 0 & & & \\
 C \rightarrow 1 & & & \\
 \dots & & & \\
 C \rightarrow 9 & | & &
 \end{array}$$

2 Assembleur

Soit le programme L suivant :

```
somme(entier $x, entier $y) { retour $x + $y;}  
main()entier $x; {$x = 1; somme($x, 2);}
```

Exercice 2.1 Dessinez l'arbre abstrait correspondant au programme.

Voici la liste de certains types de nœuds de l'arbre abstrait

appel	appel de fonction
foncDec	déclaration de fonction
instr_X	instruction de type X
XExp	expression de type X
l_dec	liste de déclarations
l_exp	liste d'expressions
l_instr	liste d'instructions
prog	programme
varDec	déclaration de variable
var_simple	variable simple

```
<prog>  
  <l_dec>  
    <foncDec>  
      somme  
      <l_dec>  
        <varDec>$x</varDec>  
        <l_dec>  
          <varDec>$y</varDec>  
        </l_dec>  
      </l_dec>  
    <l_instr>  
      <instr_retour>  
        <opExp>  
          plus  
          <varExp>  
            <var_simple>$x</var_simple>  
          </varExp>  
          <varExp>  
            <var_simple>$y</var_simple>  
          </varExp>  
        </opExp>  
      </instr_retour>  
    </l_instr>  
  </foncDec>  
  <l_dec>  
    <foncDec>  
      main  
      <l_dec>  
        <varDec>$x</varDec>  
      </l_dec>  
    <l_instr>  
      <instr_affect>  
        <var_simple>$x</var_simple>  
        <intExp>1</intExp>  
      </instr_affect>  
    <l_instr>  
      <instr_appel>
```

```

    <appel>
      somme
      <l_exp>
        <varExp>
          <var_simple>$x</var_simple>
        </varExp>
      </l_exp>
      <intExp>2</intExp>
      <l_exp>
        </l_exp>
      </l_exp>
    </l_exp>
  </instr_appel>
</l_instr>
</foncDec>
</l_dec>
</l_dec>
</prog>

```

Exercice 2.2 Ecrivez le programme en assembleur X86 correspondant au programme, que génèrerait votre compilateur. Chaque ligne doit être associée à un commentaire décrivant son rôle.

Voici les premières lignes du code généré :

```

#include 'io.asm'
section .bss
sinput: resb 255 ;reserve a 255 byte space in memory for the users input string
section .text
global _start
_start:
call main
mov eax, 1 ; 1 est le code de SYS_EXIT
int 0x80 ; exit

```

Voici la liste des registres que vous avez besoin de connaître :

esp	adresse du sommet de pile
ebp	adresse de base de l'espace global
eip	adresse de la prochaine instruction à exécuter
eax, ebx, ecx, edx	registres généraux

Et voici la liste (minimale) des instructions dont vous avez besoin :

add	dest	src	effectue dest = dest + src
sub	dest	src	effectue dest = dest - src
mov	dest	src	copie src dans dest
push	src		copie src au sommet de la pile
pop	dest		copie les 4 octets du sommet de la pile dans dest
call	adr		empile eip et va à l'adresse adr
ret			depile eip

```

#include 'io.asm'

```

```

section .bss
sinput: resb 255 ;reserve a 255 byte space in memory for the users input string

```

```

section .text
global _start
_start:
call main
mov eax, 1 ; 1 est le code de SYS_EXIT
int 0x80 ; exit
somme:
push ebp ; sauvegarde la valeur de ebp
mov ebp, esp ; nouvelle valeur de ebp
mov ebx, [ebp + 12] ; lit variable dans ebx
push ebx
mov ebx, [ebp + 8] ; lit variable dans ebx
push ebx
pop ebx ; depile la seconde operande dans ebx
pop eax ; depile la première operande dans eax
add eax, ebx ; effectue l'opération
push eax ; empile le résultat
pop eax
mov [ebp + 16], eax ; ecriture de la valeur de retour
pop ebp ; restaure la valeur de ebp
ret
pop ebp ; restaure la valeur de ebp
ret
main:
push ebp ; sauvegarde la valeur de ebp
mov ebp, esp ; nouvelle valeur de ebp
sub esp, 4 ; allocation variables locales
push 1
pop ebx
mov [ebp - 4], ebx ; stocke registre dans variable
sub esp, 4 ; allocation valeur de retour
; empile arg 0
mov ebx, [ebp - 4] ; lit variable dans ebx
push ebx
; empile arg 1
push 2
call somme
add esp, 8 ; desallocation parametres
add esp, 4 ; valeur de retour ignoree
add esp, 4 ; desallocation variables locales
pop ebp ; restaure la valeur de ebp
ret

```