Faculté des Sciences
Aix*Marseille Université

Année universitaire 2016/2017

Site : ☒ Luminy ☐ St-Charles ☐ St-Jérôme ☐ Cht-Gombert ☒ Aix-Montperrin ☐ Aubagne-SATIS
Sujet de : ☐ 1er semestre ☒ 2ème 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

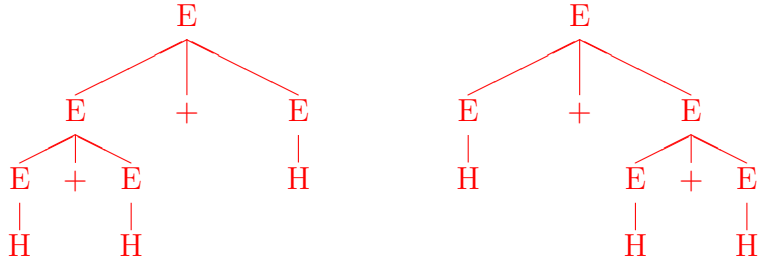
---

1 Chemins

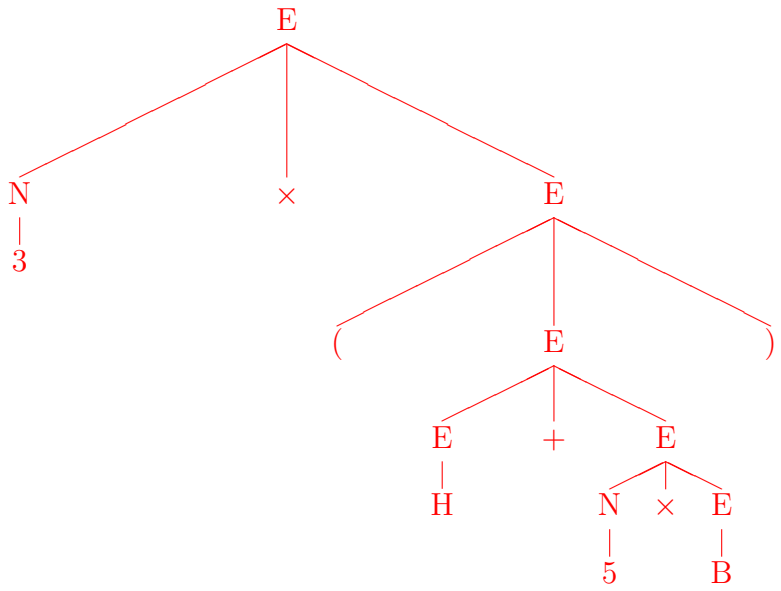
On considère un point dans le plan, identifié par ses coordonnées cartésiennes (x, y). Ce point est initialement situé aux coordonnées $(0, 0)$ et peut effectuer quatre mouvements élémentaires d'une unité, vers le haut, le bas, la gauche et la droite. A chacun de ces mouvements correspond une lettre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

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		...	
E	$\rightarrow$	G	C	$\rightarrow$	9

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 $L(G_1) = L(G_0)$

$$
\begin{array}{lcl}
E & \to & E + E' \\
E & \to & E' \\
E' & \to & N \times E' \\
E' & \to & (E) \\
E' & \to & H \\
E' & \to & B \\
E' & \to & G
\end{array}
\qquad
\begin{array}{lcl}
E' & \to & D \\
N & \to & NC \\
N & \to & C \\
C & \to & 0 \\
 & \cdots & \\
C & \to & 9
\end{array}
$$

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

$$
\begin{array}{lcl}
E & \to & E'E'' \\
E'' & \to & +E'E'' \\
E'' & \to & \varepsilon \\
E' & \to & N \times E' \\
E' & \to & (E) \\
E' & \to & H \\
E' & \to & B \\
E' & \to & G
\end{array}
\qquad
\begin{array}{lcl}
E' & \to & D \\
N & \to & CN \\
N & \to & C \\
C & \to & 0 \\
 & \cdots & \\
C & \to & 9
\end{array}
$$

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

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

Question 1.6 Calculer les premiers et suivants des symboles de G_3

	$PREMIERS$	$SUIVANTS$
E	$0,1,2,3,4,5,6,7,8,9,(,H,B,G,D$	$\perp,)$
E'	$0,1,2,3,4,5,6,7,8,9,(,H,B,G,D$	$+,\perp,)$
E''	$+,\varepsilon$	$\perp,)$
N	$0,1,2,3,4,5,6,7,8,9$	$\times$
N'	$0,1,2,3,4,5,6,7,8,9$	$\times$
C	$0,1,2,3,4,5,6,7,8,9$	$0,1,2,3,4,5,6,7,8,9$

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.

	$+$	$\times$	$($	$)$	G	H	B	D	0	$\ldots$	9	$\perp$
E					1	1	1	1	1	1	1	
E'			5		8	6	7	9	4	4	4	
E''	2			3								3
N									10	10	10	
N'		12							11	11	11	
C									13	$\ldots$	22	

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

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

E	$\to$	$E + E$	$E_0.M = E_1.M + E_2.M$
E	$\to$	$N \times E$	$E_0.M = N.M \times E_1.M$
E	$\to$	(E)	$E_0.M = E_1.M$
E	$\to$	H	$E_0.M = 1$
E	$\to$	B	$E_0.M = 1$
E	$\to$	G	$E_0.M = 1$
E	$\to$	D	$E_0.M = 1$
N	$\to$	NC	$N_0.M = 10 \times N_1.M + C.M$
N	$\to$	C	$N_0.M = C.M$
C	$\to$	0	$C_0.M = 0$
C	$\to$	1	$C_0.M = 1$
$\dots$			
C	$\to$	9	$C_0.M = 9$

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

E	$\to$	$E + E$	$E_0.x = E_1.x + E_2.x$	$E_0.y = E_1.y + E_2.y$
E	$\to$	$N \times E$	$E_0.x = N.M \times E_1.x$	$E_0.y = N.M \times E_1.y$
E	$\to$	(E)	$E_0.x = E_1.x$	$E_0.y = E_1.y$
E	$\to$	H	$E_0.x = 1$	$E_0.y = 0$
E	$\to$	B	$E_0.x = -1$	$E_0.y = 0$
E	$\to$	G	$E_0.x = 0$	$E_0.y = -1$
E	$\to$	D	$E_0.x = 0$	$E_0.y = 1$
N	$\to$	NC		
N	$\to$	C		
C	$\to$	0		
C	$\to$	1		
$\dots$				
C	$\to$	9		

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 herité x et l'attribut synthétisé m. x représente l'abscisse courant et m l'abscisse maximal. On ajoute la règle $S \to E$ afin d'initialiser l'attribut x.

S	$\to$	E	$E_1.x = 0$		
E	$\to$	$E + E$	$E_1.x = E_0.x$	$E_2.x = E_1.m$	$E_0.m = \max(E_1.m, E_2.m)$
E	$\to$	$N \times E$	$E_1.x = E_0.x$		$E_0.m = N_M * E_1.m$
E	$\to$	(E)	$E_1.x = E_0.x$		
E	$\to$	H			$E_0.m = E_0.x + 1$
E	$\to$	B			$E_0.m = E_0.x - 1$
E	$\to$	G			$E_0.m = E_0.x$
E	$\to$	D			$E_0.m = E_0.x$
N	$\to$	NC			
N	$\to$	C			
C	$\to$	0			
C	$\to$	1			
$\dots$					
C	$\to$	9			

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>
              </appel>
            </instr_appel>
          </l_instr>
        </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 sont 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 permière 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
```