UNIVERSITÉ PARIS-EST MARNE-LA-VALLÉE

Examen final du 29 mai 2019

Compilation

—Licence d'informatique - 3e année—

Durée : 2 h. Les quatre parties sont indépendantes. Documents papier autorisés. Calculatrices interdites. Le barème donné ci-dessous est indicatif.

▸ Exercice 1. *Expressions de type en C (4 points)*

On représente les expressions de type en C à l'aide des types simples (`int`, `float`*...), de* `void` *et de quatre constructeurs de type :*

- `pointeur(`t`)`, *pointeur sur* t *;*
- `tableau(`x`..`y`,` t`)`, *tableau de* t *avec des indices variant de* x *à* y *;*
- `fonction(`p`,`t`)`, *fonction qui prend des paramètres de type* p *et renvoie un* t*.*
- $p \times q$, *pour combiner deux types* p *et* q*.*

Pour chacune des déclarations suivantes, donnez

- *le parenthésage complet du déclarateur ;*
- *l'expression de type de la variable déclarée, soit en écrivant une formule comme* `tableau(0..255,pointeur(char))`*, soit en dessinant un arbre.*

Exemple :

```
char * string_list[256] ;
```

- `char (* (string_list[256])) ;`
- `tableau(0..255,pointeur(char))`

1. `int (* lengths)[512] ;`
2. `int * (* find_pivot)() ;`
3. `float * * (* * chooser)(char, float) ;`
4. `int (* find_median)(* int, * int) ;`

▸ Exercice 2. *Grammaires attribuées (5 points)*

On utilise la grammaire algébrique ci-dessous pour représenter les nombres écrits en chiffres romains :

N	$\longrightarrow$	*inf10* \$
N	$\longrightarrow$	*inf100* \$
N	$\longrightarrow$	*inf100 inf10* \$
inf10	$\longrightarrow$	*inf4*
inf10	$\longrightarrow$	`I V`
inf10	$\longrightarrow$	`V`
inf10	$\longrightarrow$	`V` *inf4*
inf10	$\longrightarrow$	`I X`
inf4	$\longrightarrow$	`I`
inf4	$\longrightarrow$	`I I`
inf4	$\longrightarrow$	`I I I`
inf100	$\longrightarrow$	*inf40*
inf100	$\longrightarrow$	`X L`
inf100	$\longrightarrow$	`L`
inf100	$\longrightarrow$	`L` *inf40*
inf100	$\longrightarrow$	`X C`
inf40	$\longrightarrow$	`X`
inf40	$\longrightarrow$	`X X`
inf40	$\longrightarrow$	`X X X`

Ajoutez des attributs à la grammaire et insérez des actions pour obtenir une grammaire attribuée qui calcule la valeur des nombres.

▸ **Exercice 3.** ***Génération de code en bison (7 points)***

Le langage de programmation Logo permet de dessiner. On demande d'écrire un compilateur qui accepte du code Logo simplifié, et produit du code qui dessine la même chose mais dans un autre langage.

On dessine en langage Logo en contrôlant les mouvements d'une tortue qui trace un trait le long de son trajet. Pour cela, on peut faire avancer la tortue sur une certaine distance dans la direction vers laquelle elle regarde, ou la faire tourner sur elle-même, ce qui change la direction. La tortue ne peut regarder que dans 4 directions : à droite (0), en bas (1), à gauche (2) ou en haut (3). Au début d'un programme Logo, la tortue regarde vers la droite et commence à la position $(0, 0)$*. Les commandes Logo sont les suivantes :* `FORWARD` n*,* `LEFT` *et* `RIGHT`*, où* n *est un entier qui représente la distance à parcourir. Voir la figure 1 pour du code Logo et la 2 pour le dessin associé.*

```
FORWARD 3
RIGHT
FORWARD 1
LEFT
FORWARD 1
LEFT
FORWARD 2
```

FIGURE 1 – Exemple de code Logo

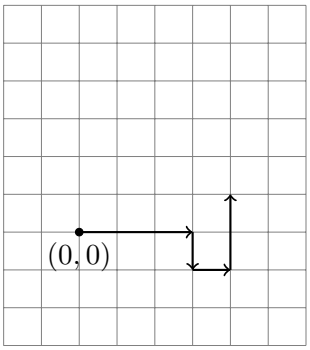

FIGURE 2 – Le dessin obtenu. Le quadrillage, le point et les flèches ne sont là qu'à titre informatif

Le langage Logo est décrit par la grammaire suivante :

$$
\begin{array}{lcl}
S & \rightarrow & suiteC \\
suiteC & \rightarrow & suiteC\ C \\
 & | & C \\
C & \rightarrow & FORWARD\ NUM \\
 & | & RIGHT \\
 & | & LEFT
\end{array}
$$

Le langage cible n'a qu'une instruction : `DRAW` $n_1\ n_2\ n_3\ n_4$ *qui trace un trait depuis le point de coordonnées* (n_1, n_2) *jusqu'au point de coordonnées* (n_3, n_4)*.*

DRAW 0 0 3 0
DRAW 3 0 3 −1
DRAW 3 −1 4 −1
DRAW 4 −1 4 1

FIGURE 3 – Le fichier obtenu après la compilation

Indication : lisez les 4 questions ci-dessous avant de commencer. Vous pouvez répondre en une seule fois à 2 ou 3 questions.

1. *Dessinez l'arbre de dérivation de la séquence de la figure 1.*
2. *Écrivez en* `bison` *un compilateur qui lit du code Logo et écrit du code équivalent dans le langage cible. Votre compilateur affichera le code résultat sur le terminal. Par exemple, votre programme doit pouvoir lire le code de la figure 1 et écrire celui de la 3. On ne demande pas d'écrire un programme* `flex` *: vous pouvez supposer qu'il existe.*
3. *Le langage Logo permet maintenant de donner une couleur au trait avec la commande* `COLOUR` *n, où n est un entier représentant une couleur. Tous les traits tracés après cette commande seront de cette couleur jusqu'à ce qu'on rechange de couleur. Au début les traits sont en noir (la valeur* 0*). L'instruction* `DRAW` *prend un* 5^e *argument entier qui représente la couleur du trait.*

   Apportez toutes les modifications nécessaires à votre code pour traduire cette nouvelle commande.
4. *On donne maintenant la possibilité à la tortue de décoller, ce qui permet de la déplacer sans tracer de trait. Elle peut atterrir pour recommencer à dessiner. Ces deux actions sont faites par les commandes* `PENUP` *et* `PENDOWN` *respectivement. Décoller plusieurs fois de suite ou atterrir sans avoir décollé n'ont aucun effet particulier.*

   Apportez toutes les modifications nécessaires à votre code pour traduire ces nouvelles commandes.

▸ **Exercice 4.** ***Programmation en nasm 64 bits (4 points)***
Complétez en assembleur `nasm` *64 bits le programme suivant :*

```
; expand32.asm
section .data
   source: dd 345, 3, 12, 728, 728, 730
section .bss
   destination: resq 6
section .text
global _start

_start:

    ; à compléter

    mov rax,60
    mov rdi, 0
```

```
syscall
```

Votre programme doit accéder aux 6 entiers stockés à partir de l'adresse `source` *(ligne 3), et les copier à partir de l'adresse* `destination` *(ligne 5) sur 64 bits chacun. On demande que le programme soit :*

- *indépendant du nombre d'entiers (s'il y en a 100 au lieu de 6, on doit pouvoir adapter le programme juste en changeant les lignes 3 et 5 et en remplaçant des 6 par des 100),*
- *et indépendant des valeurs de ces entiers, tant que ce sont tous des entiers positifs ou nuls.*