Université de Caen **Année 2017/2018**
UFR des Sciences **1ère session**
Lundi 19 mars

L3 – INF6ACT
Théorie des langages et compilation
durée 2h
Les notes de cours et TD sont autorisées.

Chaque candidat doit, en début d'épreuve, porter son nom dans le coin de la copie réservé à cet usage; il le cachettera par collage après la signature de la feuille d'émargement. Sur chacune des copies intercalaires, il portera son numéro de place.

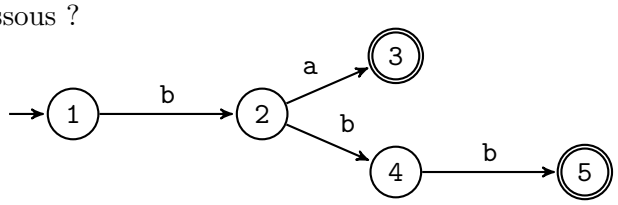
Rendre 2 copies séparées en notant bien le numéro de place :

- l'une qui traite l'exercice I (Automate fini) et l'exercice III (analyse LL)
- l'autre qui traite l'exercice II (Compilation)

Exercice I. Automate Fini — À rendre avec l'exercice III

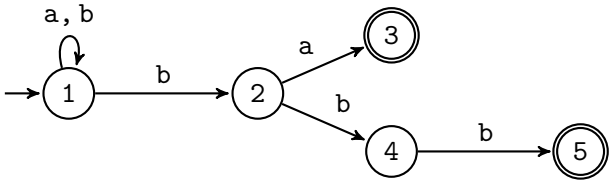
Question 1. Quel est le langage reconnu par l'automate fini déterministe $\mathcal{A}$ décrit par le graphe de transition ci-dessous ?

L'AFD $\mathcal{A}$



Question 2. On considère l'automate $\mathcal{B}$ représenté par le graphe de transition suivant :

L'AFN $\mathcal{B}$



2 . a) Pourquoi l'automate $\mathcal{B}$ est non déterministe ?

2 . b) Donner 3 mots de longueur 3 acceptés par $\mathcal{B}$ et 2 mots de longueur 4 rejetés par $\mathcal{B}$.

2 . c) Donner une expression régulière du langage reconnu par l'automate $\mathcal{B}$.

Question 3. Donner la table de transition de l'automate $\mathcal{B}$. Déterminiser cet automate.

Question 4. Donner une grammaire qui engendre le langage reconnu par l'automate $\mathcal{B}$.

Exercice II. Compilation — À rendre sur une copie séparée

On cherche à ajouter l'opérateur ternaire `?:` au langage de la calculette. Pour cela on ajoute à la règle `expression` la syntaxe suivante :

```
expression returns [String code]
        <...>
        | '(' condition '?' e1=expression ':' e2=expression ')'
        {
          // À compléter
        }
        ;
```

La sémantique associée est la suivante : si la condition est vraie alors la valeur de l'expression `e1` est utilisée; sinon, on utilise la valeur de l'expression `e2`

Soit le code suivant

```
 JUMP 0
LABEL 1
  PUSHL -3
  PUSHI 1
  INF
  JUMPF 3
  PUSHI 1
  JUMP 2
LABEL 3
  PUSHL -3
  PUSHI 0
  PUSHL -3
  PUSHI 1
  SUB
  CALL 1
  POP
  MUL
LABEL 2
  STOREL -4
  RETURN
RETURN
LABEL 0
  PUSHI 0
  PUSHI 2
  CALL 1
  POP
  WRITE
  POP
  HALT
```

et le résultat de son assemblage

```
 Adr |  Instruction
-----+---------------
   0 | JUMP       30
   2 | PUSHL      -3
   4 | PUSHI       1
   6 | INF
   7 | JUMPF      13
   9 | PUSHI       1
  11 | JUMP       26
  13 | PUSHL      -3
  15 | PUSHI       0
  17 | PUSHL      -3
  19 | PUSHI       1
  21 | SUB
  22 | CALL        2
  24 | POP
  25 | MUL
  26 | STOREL     -4
  28 | RETURN
  29 | RETURN
  30 | PUSHI       0
  32 | PUSHI       2
  34 | CALL        2
  36 | POP
  37 | WRITE
  38 | POP
  39 | HALT
```

Question 5. Compléter la trace d'exécution suivante.

```
  pc |              |    fp   pile
=================================================
   0 | JUMP       30 |     0 [ ] 0
  30 | PUSHI       0 |     0 [ ] 0
  32 | PUSHI       2 |     0 [ 0 ] 1
  34 | CALL        2 |     0 [ 0 2 ] 2
   2 | PUSHL      -3 |     4 [ 0 2 36 0 ] 4
   4 | PUSHI       1 |     4 [ 0 2 36 0 2 ] 5
   6 | INF           |     4 [ 0 2 36 0 2 1 ] 6
   7 | JUMPF      13 |     4 [ 0 2 36 0 0 ] 5
  13 | PUSHL      -3 |     4 [ 0 2 36 0 ] 4
  15 | PUSHI       0 |     4 [ 0 2 36 0 2 ] 5
  17 | PUSHL      -3 |     4 [ 0 2 36 0 2 0 ] 6
  19 | PUSHI       1 |     4 [ 0 2 36 0 2 0 2 ] 7
  21 | SUB           |     4 [ 0 2 36 0 2 0 2 1 ] 8
  22 | CALL        2 |     4 [ 0 2 36 0 2 0 1 ] 7
     |               |
     |               |
     |               |
     |               |
     |               |
  15 | PUSHI       0 |     9 [ 0 2 36 0 2 0 1 24 4 1 ] 10
  17 | PUSHL      -3 |     9 [ 0 2 36 0 2 0 1 24 4 1 0 ] 11
  19 | PUSHI       1 |     9 [ 0 2 36 0 2 0 1 24 4 1 0 1 ] 12
  21 | SUB           |     9 [ 0 2 36 0 2 0 1 24 4 1 0 1 1 ] 13
  22 | CALL        2 |     9 [ 0 2 36 0 2 0 1 24 4 1 0 0 ] 12
   2 | PUSHL      -3 |    14 [ 0 2 36 0 2 0 1 24 4 1 0 0 24 9 ] 14
   4 | PUSHI       1 |    14 [ 0 2 36 0 2 0 1 24 4 1 0 0 24 9 0 ] 15
   6 | INF           |    14 [ 0 2 36 0 2 0 1 24 4 1 0 0 24 9 0 1 ] 16
   7 | JUMPF      13 |    14 [ 0 2 36 0 2 0 1 24 4 1 0 0 24 9 1 ] 15
   9 | PUSHI       1 |    14 [ 0 2 36 0 2 0 1 24 4 1 0 0 24 9 ] 14
  11 | JUMP       26 |    14 [ 0 2 36 0 2 0 1 24 4 1 0 0 24 9 1 ] 15
  26 | STOREL     -4 |    14 [ 0 2 36 0 2 0 1 24 4 1 0 0 24 9 1 ] 15
  28 | RETURN        |    14 [ 0 2 36 0 2 0 1 24 4 1 1 0 24 9 ] 14
  24 | POP           |     9 [ 0 2 36 0 2 0 1 24 4 1 1 0 ] 12
     |               |
     |               |
     |               |
     |               |
  25 | MUL           |     4 [ 0 2 36 0 2 1 ] 6
  26 | STOREL     -4 |     4 [ 0 2 36 0 2 ] 5
  28 | RETURN        |     4 [ 2 2 36 0 ] 4
  36 | POP           |     0 [ 2 2 ] 2
  37 | WRITE         |     0 [ 2 ] 1
2
  38 | POP           |     0 [ 2 ] 1
  39 | HALT          |     0 [ ] 0
```

Question 6. Donner le code MVaP généré par le programme suivant :

```
int x = 1
println(1 + ( x > 2 ? 4 : x))
```

Question 7. Compléter le code antlr pour obtenir la génération de code.

Question 8. Expliquer ce que fait le code MVaP donné en début d'exercice. Donner un programme produisant ce code.

Exercice III. Analyse LL — À rendre avec l'exercice I

On considère la grammaire `G` qui engendre l'ensemble des mots qui ont autant de `a` que de `b`. Elle est définie par l'axiome `S`, les variables {`S, C, D`}, les terminaux {`a, b`} et les règles de production :

$$\begin{cases} \texttt{S} \rightarrow \texttt{aDbS} \mid \texttt{bCaS} \mid \varepsilon \\ \texttt{D} \rightarrow \texttt{aDbD} \mid \varepsilon \\ \texttt{C} \rightarrow \texttt{bCaC} \mid \varepsilon \end{cases}$$

Question 9. Donner pour le mot `bbaaab` un arbre d'analyse et la dérivation gauche associée.

On dispose des tables `Effacable`, `Premier` et `Suivant` de la grammaire `G` :

Symbole	Effacable	Premier	Suivant
S	Oui	a, b	$
D	Oui	a	b
C	Oui	b	a

Question 10. Indiquer comment est déterminé l'ensemble `Premier(S)` et l'ensemble `Suivant(C)`.

Question 11. Construire la table d'analyse `LL(1)` de la grammaire `G`. Qu'est-ce qui permet d'affirmer que cette grammaire est `LL(1)` ?

Question 12.

12.a) Dérouler l'analyse `LL(1)` sur l'entrée `bbaaab`. À partir de cette analyse, comment retrouver la dérivation gauche associée à `bbaaab` ?

12.b) Dérouler l'analyse `LL(1)` sur l'entrée `aba`.