Exercice 1 (Syphonage de page web).

1. Ecrivez une expression rationnelle reconnaissant une adresse mail sur les domaines .com, .fr ou .org uniquement.

↪ $\text{alphanumeric}^* \text{'@'} \text{alphanumeric}^* \text{'.'} ("com" | "fr" | "org")$

où $\text{alphanumeric} = [\text{'a'}-\text{'z'}, \text{'0'}-\text{'9'}, \text{'.'}, \text{'-'}, \text{'_'}]$ est une macro

2. Donner l'automate non-déterministe correspondant.

↪ dessin à venir, en attendant voici la description :

$$Q := \{q_0, q_{nom}, q_{@}, q_{dom}, q_{.}, q_c, q_{co}, q_{com}, q_f, q_{fr}, q_o, q_{or}, q_{org}\}$$

$$F = \{q_{com}, q_{fr}, q_{org}\}$$

$$\tau(q_0, c) = q_{nom} \text{ pour tout } c \in \{\text{'a'}-\text{'z'}, \text{'0'}-\text{'9'}, \text{'.'}, \text{'-'}, \text{'_'}\}$$

$$\tau(q_{nom}, c) = q_{nom} \text{ pour tout } c \in \{\text{'a'}-\text{'z'}, \text{'0'}-\text{'9'}, \text{'.'}, \text{'-'}, \text{'_'}\}$$

$$\tau(q_{nom}, \text{'@'}) = q_{@}$$

$$\tau(q_{@}, c) = q_{dom} \text{ pour tout } c \in \{\text{'a'}-\text{'z'}, \text{'0'}-\text{'9'}, \text{'.'}, \text{'-'}, \text{'_'}\}$$

$$\tau(q_{dom}, c) = q_{dom} \text{ pour tout } c \in \{\text{'a'}-\text{'z'}, \text{'0'}-\text{'9'}, \text{'.'}, \text{'-'}, \text{'_'}\}$$

$$\tau(q_{dom}, \text{'.'}) = q_{.}$$

$$\tau(q_{.}, \text{'c'}) = q_c$$

$$\tau(q_{.}, \text{'f'}) = q_f$$

$$\tau(q_{.}, \text{'o'}) = q_o$$

$$\tau(q_c, \text{'o'}) = q_{co}$$

$$\tau(q_{co}, \text{'m'}) = q_{com}$$

$$\tau(q_f, \text{'r'}) = q_{fr}$$

$$\tau(q_o, \text{'r'}) = q_{or}$$

$$\tau(q_g, \text{'g'}) = q_{org}$$

3. Donner l'automate déterministe correspondant.

↪ dessin à venir, en attendant voici la description :

$$Q := \{q_0, q_{nom}, q_{@}, q_{dom}, q_{.}, q_c, q_{co}, q_{com}, q_f, q_{fr}, q_o, q_{or}, q_{org}\}$$

$$F = \{q_{com}, q_{fr}, q_{org}\}$$

$\tau(q_0, c) = q_{nom}$	pour tout $c \in \{'a'-'z', '0'-'9', ' ', ' ', ' ', ' ', ' ', ' ', ' '\}$
$\tau(q_{nom}, c) = q_{nom}$	pour tout $c \in \{'a'-'z', '0'-'9', ' ', ' ', ' ', ' ', ' ', ' ', ' '\}$
$\tau(q_{nom}, '@') = q_{@}$	
$\tau(q_{@}, c) = q_{dom}$	pour tout $c \in \{'a'-'z', '0'-'9', ' ', ' ', ' ', ' ', ' ', ' ', ' '\}$
$\tau(q_{dom}, c) = q_{dom}$	pour tout $c \in \{'a'-'z', '0'-'9', ' ', ' ', ' ', ' ', ' ', ' ', ' '\}$
$\tau(q_{dom}, '.') = q_{\{dom,.\}}$	
$\tau(q_{\{dom,.\}}, c) = q_{dom}$	pour tout $c \in \{'a', 'b', 'd', 'e', 'g', 'n', 'p', 'z', '0', '9', ' ', ' ', ' ', ' '\}$
$\tau(q_{\{dom,.\}}, '.') = q_{\{dom,.\}}$	
$\tau(q_{\{dom,.\}}, 'c') = q_{\{dom,c\}}$	
$\tau(q_{\{dom,.\}}, 'f') = q_{\{dom,f\}}$	
$\tau(q_{\{dom,.\}}, 'o') = q_{\{dom,o\}}$	
$\tau(q_{\{dom,c\}}, 'o') = q_{\{dom,co\}}$	
$\tau(q_{\{dom,c\}}, c) = q_{dom}$	pour tout $c \in \{'a'-'n', 'p', 'z', '0', '9', ' ', ' ', ' ', ' '\}$
$\tau(q_{\{dom,c\}}, '.') = q_{\{dom,.\}}$	
$\tau(q_{\{dom,co\}}, 'm') = q_{\{dom,com\}}$	
$\tau(q_{\{dom,co\}}, c) = q_{dom}$	pour tout $c \in \{'a'-'l', 'n', 'z', '0', '9', ' ', ' ', ' ', ' '\}$
$\tau(q_{\{dom,co\}}, '.') = q_{\{dom,.\}}$	
$\tau(q_{\{dom,com\}}, c) = q_{dom}$	pour tout $c \in \{'a'-'z', '0', '9', ' ', ' ', ' ', ' '\}$
$\tau(q_{\{dom,com\}}, '.') = q_{\{dom,.\}}$	
$\tau(q_{\{dom,f\}}, 'r') = q_{\{dom,fr\}}$	
$\tau(q_{\{dom,f\}}, c) = q_{dom}$	pour tout $c \in \{'a'-'q', 's', 'z', '0', '9', ' ', ' ', ' ', ' '\}$
$\tau(q_{\{dom,f\}}, '.') = q_{\{dom,.\}}$	
$\tau(q_{\{dom,fr\}}, c) = q_{dom}$	pour tout $c \in \{'a'-'z', '0', '9', ' ', ' ', ' ', ' '\}$
$\tau(q_{\{dom,fr\}}, '.') = q_{\{dom,.\}}$	
$\tau(q_{\{dom,o\}}, 'r') = q_{\{dom,or\}}$	
$\tau(q_{\{dom,o\}}, c) = q_{dom}$	pour tout $c \in \{'a'-'q', 's', 'z', '0', '9', ' ', ' ', ' ', ' '\}$
$\tau(q_{\{dom,o\}}, '.') = q_{\{dom,.\}}$	
$\tau(q_{\{dom,or\}}, 'g') = q_{\{dom,org\}}$	
$\tau(q_{\{dom,or\}}, c) = q_{dom}$	pour tout $c \in \{'a'-'f', 'h', 'z', '0', '9', ' ', ' ', ' ', ' '\}$
$\tau(q_{\{dom,or\}}, '.') = q_{\{dom,.\}}$	
$\tau(q_{\{dom,org\}}, c) = q_{dom}$	pour tout $c \in \{'a'-'z', '0', '9', ' ', ' ', ' ', ' '\}$
$\tau(q_{\{dom,org\}}, '.') = q_{\{dom,.\}}$	

4. Ecrivez un transducteur à buffer (déterministe ou à priorité) donnant la liste des mots important d'une page .html. On considère qu'un mot est important s'il fait plus de 3 caractères et l'un des trois : apparait dans un titres ou est en italique.

↪ dessin à venir, en attendant voici la description ; il n'y a pas de priorité car il est déterministe :

$$Q := \{q_0, q_<, q_<h, q_<hi, q_{t0}, q_{t1}, q_{t2}, q_{t3}, q_{t<}, q_{t<\}, q_{t<\}h, q_{t<\}hi}, \\ q_{<e}, q_{<em}, q_i, q_{i<}, q_{i<\}, q_{i<\}e}, q, q_o, q_{or}, q_{org}\}$$

$$F = \{q_0\}$$

$\tau(q_0, ' <') = ((), (), q_<)$		$\tau(q_0, c) = ((), (), q_0)$	sinon	
$\tau(q_<, ' h') = ((), (), q_<h)$		$\tau(q_<, ' e') = ((), (), q_<e)$		
$\tau(q_<, c) = ((), (), q_0)$	sinon	$\tau(q_<h, i) = ((), (), q_<hi)$	si $i \in \{1, 2, 3, 4, 5\}$	
$\tau(q_<h, c) = ((), (), q_0)$	sinon	$\tau(q_<hi, ' >') = ((), \mathbf{flush}, q_{t1})$		
$\tau(q_<, c) = ((), (), q_0)$	sinon	$\tau(q_{t0}, ' <') = ((), (), q_{t<})$		
$\tau(q_{t0}, c) = ((), (), q_{t1})$	si $c \in \{ ' a' - ' Z' \}$	$\tau(q_{t0}, c) = ((), \mathbf{flush}, q_{t1})$	sinon	$\tau(q_{t1}, ' <') = ((), (), q_{t<})$
$\tau(q_{t1}, c) = ((), \mathbf{flush}, q_{t1})$	sinon	$\tau(q_{t2}, ' <') = ((), (), q_{t<})$		
$\tau(q_{t2}, c) = ((), (), q_{t3})$	si $c \in \{ ' a' - ' Z' \}$	$\tau(q_{t2}, c) = (\mathbf{flush}, (), q_{t1})$	sinon	
$\tau(q_{t3}, ' <') = ((), \mathbf{store}, q_{t<})$		$\tau(q_{t3}, c) = ((), (), q_{t3})$	si $c \in \{ ' a' - ' Z' \}$	
$\tau(q_{t3}, c) = (\mathbf{store}, \mathbf{flush}, q_{t1})$	sinon	$\tau(q_{t<}, ' \setminus') = ((), (), q_{t<\})$		
$\tau(q_{t<}, c) = ((), \mathbf{flush}, q_{t1})$	sinon	$\tau(q_{t<\}, ' h') = ((), (), q_{t<\}h)$		
$\tau(q_{t<\}, c) = ((), \mathbf{flush}, q_{t1})$	sinon	$\tau(q_{t<\}h, i) = ((), (), q_{t<\}hi)$	si $i \in \{1, 2, 3, 4, 5\}$	
$\tau(q_{t<\}h, c) = ((), \mathbf{flush}, q_{t1})$	sinon	$\tau(q_{t<\}hi, ' >') = ((), (), q_0)$		
$\tau(q_{t<\}hi, c) = ((), \mathbf{flush}, q_{t1})$	sinon	$\tau(q_{<e}, ' m') = ((), (), q_{<em})$		
$\tau(q_{<e}, c) = ((), (), q_0)$	sinon	$\tau(q_{<em}, ' >') = ((), \mathbf{flush}, q_i)$		
$\tau(q_{<em}, c) = ((), (), q_0)$	sinon	$\tau(q_i, ' <') = (\mathbf{store}, \mathbf{flush}, q_{i<})$		
$\tau(q_i, c) = ((), (), q_i)$	si $c \in \{ ' a' - ' Z' \}$	$\tau(q_i, c) = (\mathbf{store}, \mathbf{flush}, q_i)$	sinon	
$\tau(q_{i<}, ' \setminus') = ((), (), q_{i<\})$		$\tau(q_{i<}, c) = ((), \mathbf{flush}, q_i)$	sinon	
$\tau(q_{i<\}, ' e') = ((), (), q_{i<\}e)$		$\tau(q_{i<\}, c) = ((), \mathbf{flush}, q_i)$	sinon	
$\tau(q_{i<\}e, ' m') = ((), (), q_{i<\}em)$		$\tau(q_{i<\}e, c) = ((), \mathbf{flush}, q_i)$	sinon	
$\tau(q_{i<\}em, ' >') = ((), (), q_0)$		$\tau(q_{i<\}em, c) = ((), \mathbf{flush}, q_i)$	sinon	

où les deux actions **flush** et **store** vont respectivement vider le buffer et renvoyer le buffer.

5. Ecrivez un transducteur à buffer donnant la liste des mots important et des adresses mail d'une page .html (avec deux types de token : EMAIL et IMPORTANT).
6. Dans l'entête d'une page HTML, on trouve des mot-clef, récupérez-les aussi dans votre transducter.
7. Récupérez aussi tous les #tags cités dans la page.
8. Implémentez l'automate dans le langage de votre choix et testez-le.

Exercice 2 (Parseur LL).

On considère la grammaire suivante : $\rightsquigarrow$ En rouge, j'indique le numéro de la règle

$$\begin{aligned}
 S &:= \langle \text{IF} \rangle E \langle \text{THEN} \rangle S \langle \text{ELSE} \rangle S^{(1)} \\
 &\quad | \langle \text{IF} \rangle E \langle \text{THEN} \rangle S^{(2)} \\
 &\quad | \langle \text{RETURN} \rangle E;^{(4)} \\
 &\quad | \langle \text{VAR} \rangle = E;^{(5)} \\
 E &:= P H^{(6)} \\
 H &:= \epsilon^{(7)} \quad | \quad == P^{(8)} \\
 P &:= T U^{(9)} \\
 T &:= F V^{(10)} \\
 U &:= \epsilon^{(11)} \quad | \quad * P^{(12)} \quad | \quad \&\& P^{(13)} \\
 V &:= \epsilon^{(14)} \quad | \quad + T^{(15)} \quad | \quad - T^{(16)} \quad | \quad || T^{(17)} \\
 F &:= \langle \text{VAR} \rangle^{(18)} \quad | \quad \langle \text{INT} \rangle^{(19)} \quad | \quad (E)^{(20)} \\
 &\quad | \quad - F^{(21)} \quad | \quad \langle \text{NON} \rangle F^{(22)}
 \end{aligned}$$

où les non-terminaux sont indiqués par des lettres majuscules et les tokens sont indiqués par des symboles ou des noms entre crochets.

- Calculer les firsts et les follows.

	$\langle \text{IF} \rangle$	$\langle \text{THEN} \rangle$	$\langle \text{ELSE} \rangle$	$\langle \text{RETURN} \rangle$	$\langle \text{VAR} \rangle$	$\langle \text{INT} \rangle$	$\langle \text{NON} \rangle$	+	-	*	&&		()	=	;	ϵ	\$
S	1,2		F	4	5												F
E		F			6	6	6		6				6	F		F	
H		F											F	8	F	7	
P		F			9	9	9		9				9	F	F	F	
T		F			10	10	10		10	F	F		10	F	F	F	
U		F								12	13			F	F	F	11
V		F						15	16	F	F	17		F	F	F	14
F		F			18	19	22	F	21,F	F	F	F	20	F	F	F	

- Cette grammaire n'est pas LL(1), pourquoi?

$\rightsquigarrow$ Car il y a deux règles dans la case $(S, \langle \text{IF} \rangle)$, on ne sera donc pas laquelle appliquer : c'est un conflit first/first.

Attention : il semble y avoir un conflit first/follow dans la case $(F, -)$, mais celui-ci n'est pas gênant car il n'y a pas de règle en (F, ϵ) .

- En fait elle est ambiguë, voyez-vous pourquoi?

$\rightsquigarrow$ Le problème arrive entre les règles (1) et (2), on peut donc les utiliser pour trouver notre contre exemple :

```
if (2==2) then if (2==3) then return 1; else return 2;
```

Ce programme est correcte selon la grammaire, mais on ne sais pas s'il s'agit d'un if-then-else dans un if-then ou d'un if-then dans un if-then-else.

4. Trouver une modification mineure régler l'ambiguïté.

↪ La solution classique est de donner la priorité au if-then-else :

$$\begin{aligned}
 S &:= S'^{(0)} \\
 &\quad | \langle \text{IF} \rangle E \langle \text{THEN} \rangle S^{(1)} \\
 &\quad | \langle \text{IF} \rangle E \langle \text{THEN} \rangle S' \langle \text{ELSE} \rangle S^{(2)} \\
 S' &:= \langle \text{IF} \rangle E \langle \text{THEN} \rangle S' \langle \text{ELSE} \rangle S'^{(3)} \\
 &\quad | \langle \text{RETURN} \rangle E;^{(4)} \\
 &\quad | \langle \text{VAR} \rangle = E;^{(5)} \\
 E &:= P H^{(6)} \\
 H &:= \epsilon^{(7)} \quad | \quad == P^{(8)} \\
 P &:= T U^{(9)} \\
 T &:= F V^{(10)} \\
 U &:= \epsilon^{(11)} \quad | \quad * P^{(12)} \quad | \quad \&\&P^{(13)} \\
 V &:= \epsilon^{(14)} \quad | \quad + T^{(15)} \quad | \quad - T^{(16)} \quad | \quad || T^{(17)} \\
 F &:= \langle \text{VAR} \rangle^{(18)} \quad | \quad \langle \text{INT} \rangle^{(19)} \quad | \quad (E)^{(20)} \\
 &\quad | \quad - F^{(21)} \quad | \quad \langle \text{NON} \rangle F^{(22)}
 \end{aligned}$$

5. Recalculer les firsts et follow.

	<IF>	<THEN>	<ELSE>	<RETURN>	<VAR>	<INT>	<NON>	+	-	*	&&		()	=	;	€	\$
S	0,1,2			4	5												F
S'	3		F	4	5												F
E		F			6	6	6		6				6	F		F	
H		F												F	8	F	7
P		F			9	9	9		9				9	F	F	F	
T		F			10	10	10		10	F	F		10	F	F	F	
U		F								12	13			F	F	F	11
V		F						15	16	F	F	17		F	F	F	14
F		F			18	19	22	F	21,F	F	F	F	20	F	F	F	

6. Cette fois-ci, est-elle LL(1)? Si ce n'est pas le cas, essayez d'autres modifications jusqu'à ce que ce soit le cas.

↪

Elle n'est toujours pas LL(1), car il y a toujours un conflit en $(S, \langle \text{IF} \rangle)$. En fait, il n'existe pas de grammaire LL(1) avec le if-then-else et le if-then. En effet, même s'il est possible de déterminer le conflit (et de le rendre LR(1)) il n'est pas possible de le rendre LL(1) (ou même LR(k)), car on ne sais que à la fin quelle règle appliquer. C'est pourquoi en LL, il faut se

passer du if-then.¹

$$\begin{aligned}
 S &:= \langle \text{IF} \rangle E \langle \text{THEN} \rangle S \langle \text{ELSE} \rangle S \quad (2) \\
 &\quad | \langle \text{RETURN} \rangle E; \quad (4) \\
 &\quad | \langle \text{VAR} \rangle = E; \quad (5) \\
 E &:= P H \quad (6) \\
 H &:= \epsilon \quad (7) \quad | \quad == P \quad (8) \\
 P &:= T U \quad (9) \\
 T &:= F V \quad (10) \\
 U &:= \epsilon \quad (11) \quad | \quad * P \quad (12) \quad | \quad \&\& P \quad (13) \\
 V &:= \epsilon \quad (14) \quad | \quad + T \quad (15) \quad | \quad - T \quad (16) \quad | \quad || T \quad (17) \\
 F &:= \langle \text{VAR} \rangle \quad (18) \quad | \quad \langle \text{INT} \rangle \quad (19) \quad | \quad (E) \quad (20) \\
 &\quad | \quad - F \quad (21) \quad | \quad \langle \text{NON} \rangle F \quad (22)
 \end{aligned}$$

	<IF>	<THEN>	<ELSE>	<RETURN>	<VAR>	<INT>	<NON>	+	-	*	&&		(	)	=	;	€	\$
S	2		F	4	5													F
E		F			6	6	6		6				6	F		F		
H		F												F	8	F	7	
P		F			9	9	9		9				9	F	F	F		
T		F			10	10	10		10	F	F		10	F	F	F		
U		F								12	13			F	F	F	11	
V		F						15	16	F	F	17		F	F	F	14	
F		F			18	19	22	F	21,F	F	F	F	20	F	F	F		

7. Ecrire l'automate LL² correspondant.

↪ On commence par réécrire le tableau en enlevant les follows sur les non-terminaux ne se réduisant pas en ϵ , en reportant la règle pour ϵ sur les autres follows et en supprimant la colonne ϵ :

	<IF>	<THEN>	<ELSE>	<RETURN>	<VAR>	<INT>	<NON>	+	-	*	&&		(	)	=	;	\$
S	2			4	5												
E					6	6	6		6				6				
H		7												7	8	7	
P					9	9	9		9				9				
T					10	10	10		10				10				
U		11								12	13			11	11	11	
V		14						15	16	14	14	17		14	14	14	
F					18	19	22		21				20				

Nommons *tab* ce tableau.

L'automate est alors donné par $p_0 = [S, \$]$, par $\tau(t, t) = \text{Read}$ pour chaque

1. C'est pourquoi il est grisé dans la description de la grammaire du projet, je ne le demande pas pour le projet.
2. aussi appelé automate LL(1) ou encore parseur LL(1) au parseur descendant

terminal t , et par $\tau(N, t) = \text{Wait}(\text{tab}(n, t))$ pour tout non-terminal N , et tout terminal t tel que $\text{tab}(n, t)$ n'est pas vide.

8. Fait un run sur un programme de votre choix.