

Commencé le	vendredi 9 juin 2023, 13:32
État	Terminé
Terminé le	vendredi 9 juin 2023, 15:05
Temps mis	1 heure 32 min
Note	14,16 sur 20,00 (70,8%)
Feedback	Moyenne : 13,12

Question 1

Non répondue

Non noté

Si une question vous semble comporter des erreurs ou imprécisions, vulgairement parlant des bugs, ne posez pas de question oralement, mais signalez-le ci-dessous en précisant :

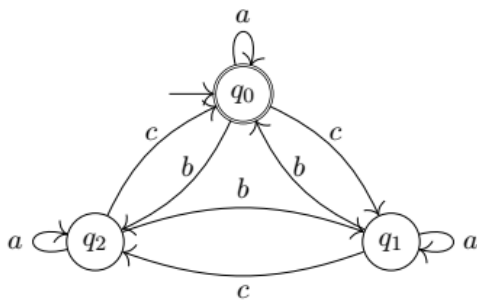
- le numéro de la question concernée
- vos interrogations sur cette question
- éventuellement l'interprétation ou les choix faits pour votre (vos) réponse(s) à cette question.

## Question 2

Correct

Note de 1,50 sur 1,50

Considérons l'AFD suivant:

(unique état acceptant:  $q_0$ , état initial:  $q_0$ )Compléter les transitions prises en lisant le mot *cacca*:

$q_0 \xrightarrow{c} q_1 \xrightarrow{a} q_1 \xrightarrow{c} q_2 \xrightarrow{c} q_0 \xrightarrow{a} q_0$

Le mot *cacca* est accepté?  ✓.Compléter les transitions prises en lisant le mot *abbac*:

$q_0 \xrightarrow{a} q_0 \xrightarrow{b} q_2 \xrightarrow{b} q_1 \xrightarrow{a} q_1 \xrightarrow{c} q_2$

Le mot *abbac* est accepté?  ✓.Compléter les transitions prises en lisant le mot *bccab*:

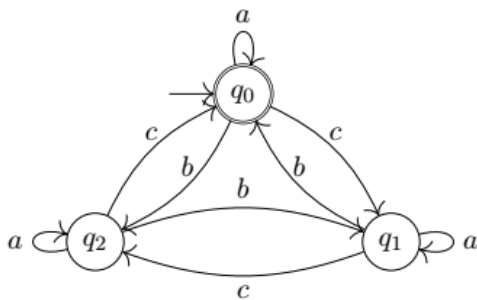
$q_0 \xrightarrow{b} q_2 \xrightarrow{c} q_0 \xrightarrow{c} q_1 \xrightarrow{a} q_1 \xrightarrow{b} q_0$

Le mot *bccab* est accepté?  ✓.

Votre réponse est correcte.

La réponse correcte est :

Considérons l'AFD suivant:

(unique état acceptant:  $q_0$ , état initial:  $q_0$ )Compléter les transitions prises en lisant le mot *cacca*:

$[q_0] \xrightarrow{c} [q_1] \xrightarrow{a} [q_1] \xrightarrow{c} [q_2] \xrightarrow{c} [q_0] \xrightarrow{a} [q_0]$

Le mot *cacca* est accepté? [oui].Compléter les transitions prises en lisant le mot *abbac*:

$[q_0] \xrightarrow{a} [q_0] \xrightarrow{b} [q_2] \xrightarrow{b} [q_1] \xrightarrow{a} [q_1] \xrightarrow{c} [q_2]$ .

Le mot *abbac* est accepté? [non].

Compléter les transitions prises en lisant le mot *bccab*:

$[q_0] \xrightarrow{b} [q_2] \xrightarrow{c} [q_0] \xrightarrow{c} [q_1] \xrightarrow{a} [q_1] \xrightarrow{b} [q_0]$ .

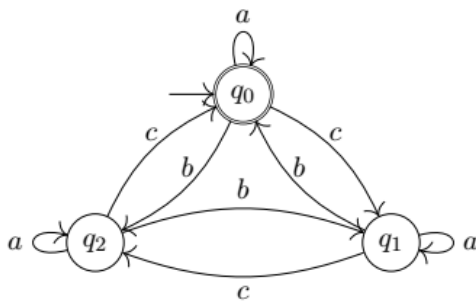
Le mot *bccab* est accepté? [oui].

### Question 3

Partiellement correct

Note de 1,33 sur 1,50

Considérons l'AFD suivant:



(unique état acceptant:  $q_0$ , état initial:  $q_0$ )

Parmi les mots suivants, sélectionner ceux reconnus par  $A$  (cliquer dessus):

☒ aaa ✓   aab   aac   aba   abb   ☒ abc ✓   aca   ☒ acb ✓   acc  
 baa   bab   ☒ bac ✓   bba   ☒ bbb ✓   bbc   ☒ bca ✓   bcb   bcc  
 caa   ☒ cab ✓   cac   ☒ cba ✓   cbb   cbc   cca   ccb   ☒ ccc ✓

Votre réponse est partiellement correcte.

**Question 4**

Correct

Note de 0,25 sur 0,25

Considérons un automate fini (déterministe ou indéterministe voir plus bas)  $A$ .

Considérons le problème de décision suivant:

- Entrée: un mot  $w$  de taille  $n$
- Sortie: Est-ce que  $w \in L$ ?

Pour les 4 questions suivantes, donner la plus petite complexité possible.

Si  $A$  est un AFD, il existe nécessairement un algorithme qui résout ce problème au pire des cas en temps:

- ☐ a.  $O(n^2)$
- ☐ b.  $O(1)$
- ☐ c.  $O(2^n)$
- ☒ d.  $O(n)$  ✓

Votre réponse est correcte.

La réponse correcte est :

$O(n)$

**Question 5**

Correct

Note de 0,25 sur 0,25

Si  $A$  est un AFD, il existe nécessairement un algorithme qui résout ce problème au pire des cas en espace:

- ☐ a.  $O(n)$
- ☐ b.  $O(n^2)$
- ☒ c.  $O(1)$  ✓
- ☐ d.  $O(2^n)$

Votre réponse est correcte.

La réponse correcte est :

$O(1)$

**Question 6**

Incorrect

Note de 0,00 sur 0,25

Si  $A$  est un AFI, il existe nécessairement un algorithme qui résout ce problème au pire des cas en temps:

- ☒ a.  $O(2^n)$  ✖
- ☐ b.  $O(1)$
- ☐ c.  $O(n^2)$
- ☐ d.  $O(n)$

Votre réponse est incorrecte.

La réponse correcte est :

$O(n)$

**Question 7**

Incorrect

Note de 0,00 sur 0,25

Si  $A$  est un AFI, il existe nécessairement un algorithme qui résout ce problème au pire des cas en espace:

- ☐ a.  $O(1)$
- ☒ b.  $O(n)$  ✖
- ☐ c.  $O(n^2)$
- ☐ d.  $O(2^n)$

Votre réponse est incorrecte.

La réponse correcte est :

$O(1)$

Question 8

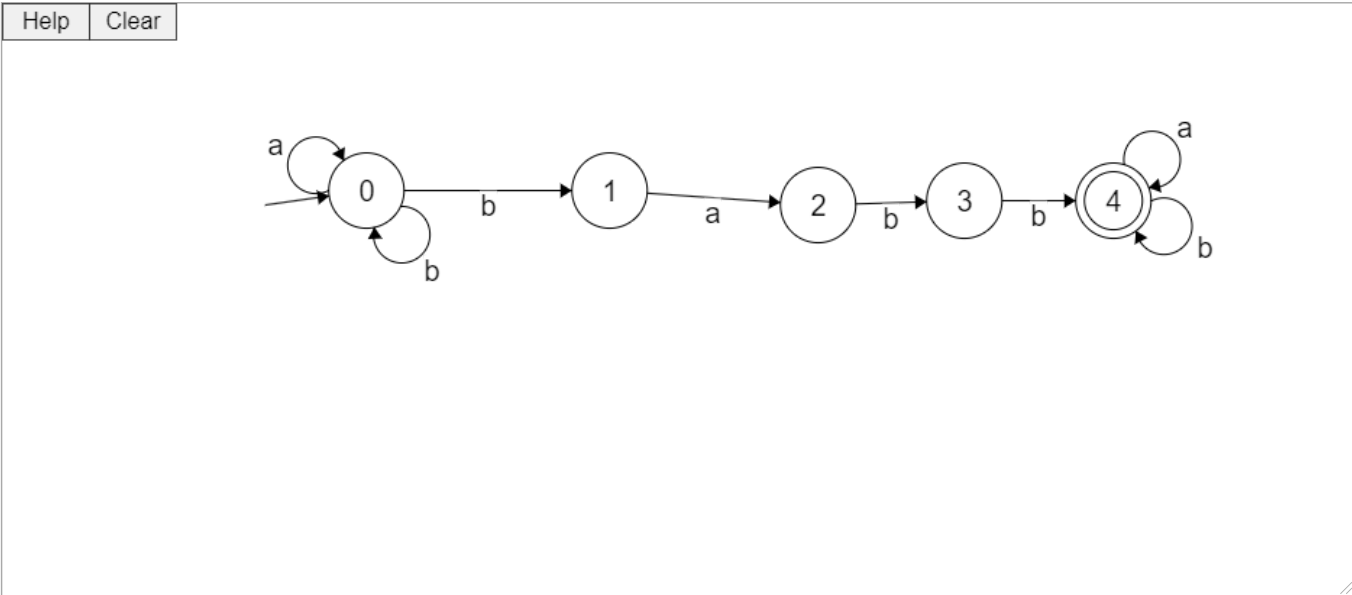
Incorrect

Note de 0,00 sur 2,00

Écrire un **AFD** qui reconnait le langage décrit par l'expression régulière :  
 $(a + b)^*babb(a + b)^*$ .

Réponse : (régime de pénalités : 0 %)

Réinitialiser la réponse



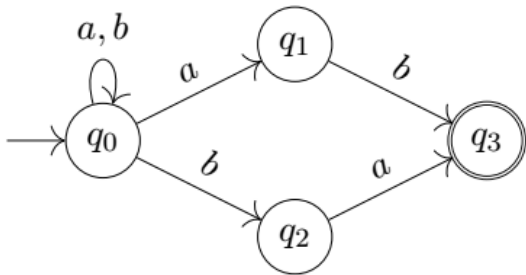
	Test	Got	Score	Total	
✖	mots de lg 0 mots de lg 1 mots de lg 2 mots de lg 3 mots de lg 4 mots de lg 5 mots de lg 6 mots de lg 7	Validation Error:Transition non-déterministe  ***Run error*** Traceback (most recent call last): File "__tester__.python3", line 41, in <module> s = set(test_mots(dfa,'ab',i)) File "__tester__.python3", line 13, in test_mots return [ "".join(x) for x in product(alphabet,repeat = longueur) if dfa.accepts_input("".join(x)) ] File "__tester__.python3", line 13, in <listcomp> return [ "".join(x) for x in product(alphabet,repeat = longueur) if dfa.accepts_input("".join(x)) ] AttributeError: 'NoneType' object has no attribute 'accepts_input'  During handling of the above exception, another exception occurred:  Traceback (most recent call last): File "__tester__.python3", line 52, in <module> raise Exception("Oops") Exception: Oops	Unknown field 'comment'	0.00	✖

► Montrer / masquer la solution de l'auteur de la question (Python3)

Incorrect

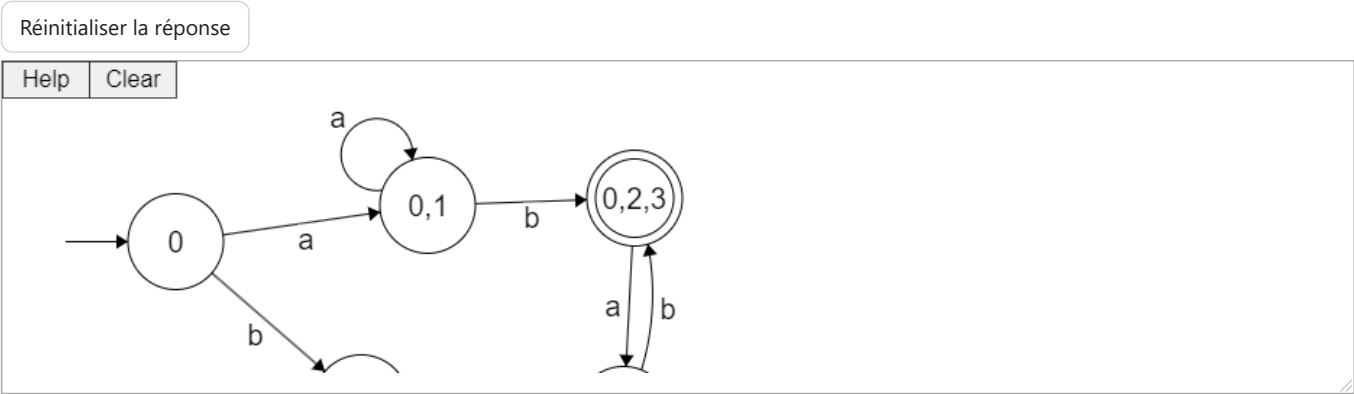
Note pour cet envoi : 0,00/2,00.

En partant de l'AFI suivant sur l'alphabet  $\{a, b\}$ :



(unique état acceptant:  $q_3$ )  
donner l'AFD obtenu par application de l'algorithme vu en cours/TD.  
(vous n'êtes pas obligé de minimiser votre AFD).  
Dans votre réponse, les numéros/noms donnés aux états ne sont pas importants.  
(votre réponse n'est pas évaluée durant le contrôle, donc toute réponse allume vert)

Réponse : (régime de pénalités : 0 %)



	Test	Résultat attendu	Résultat obtenu	
✗	dfa_s_is_complete	True	False	✗
✗	dfa_s_c_states	[0, 1, 2, 3, 4]	[0, 1, 2, 3, 4, 5]	✗
✓	dfa_s_c_accept_states	[3, 4]	[3, 4]	✓
✓	trans(0)	(1, 2)	(1, 2)	✓
✓	trans(1)	(1, 3)	(1, 3)	✓
✓	trans(2)	(4, 2)	(4, 2)	✓
✗	trans(3)	(4, 2)	(4, 5)	✗
✗	trans(4)	(1, 3)	(5, 3)	✗

Montrer les différences

Ici il s'agit de donner le résultat de l'algorithme de détermination de l'AFI donné.  
Chacun des états du DFA obtenu, est un sous-ensemble de l'AFI donné, ainsi :

- 1. **état 0** initial = ensemble des états initiaux de l'AFI donné : {0}
- 2. **état 1** = ensemble des états atteints par les différentes transitions (0,'a') dans l'AFI : {0,1}
- 3. **état 0** = ensemble des états atteints par les différentes transitions (0,'b') dans l'AFI : {0}

4. **état 1** = ensemble des états atteints par les différentes transitions  $(\{0,1\}, 'a')$  dans l'AFI :  $\{0,1\}$
5. **état 2** = ensemble des états atteints par les différentes transitions  $(\{0,1\}, 'b')$  dans l'AFI :  $\{0,2\}$
6. **état 3** = ensemble des états atteints par les différentes transitions  $(\{0,2\}, 'a')$  dans l'AFI :  $\{0,1,3\}$
7. **état 0** = ensemble des états atteints par les différentes transitions  $(\{0,2\}, 'b')$  dans l'AFI :  $\{0\}$
8. **état 1** = ensemble des états atteints par les différentes transitions  $(\{0,1,3\}, 'a')$  dans l'AFI :  $\{0,1\}$
9. **état 2** = ensemble des états atteints par les différentes transitions  $(\{0,1,3\}, 'b')$  dans l'AFI :  $\{0,2\}$

Votre réponse est correcte si et seulement si votre DFA renuméroté selon le même principe, est celui qui vient d'être construit.

On peut noter que ce DFA est **LE DFA complet minimum** qui reconnaît le langage reconnu par l'AFI donné, à savoir  $(a+b)^*aba$  (expression régulière qui se lit, sur cet exemple, plus naturellement sur l'AFI que sur le DFA).

► **Montrer / masquer la solution de l'auteur de la question (Python3)**

Partiellement correct

Note pour cet envoi : 1,00/2,00.

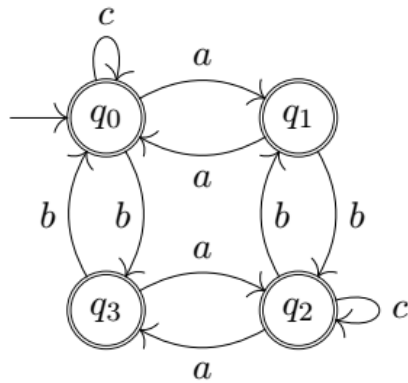


## Question 10

Correct

Note de 2,00 sur 2,00

En partant de l'automate suivant sur l'alphabet  $\{a, b, c\}$ :



( $q_0$  est l'état initial,  $q_0, q_1, q_2, q_3$  sont acceptants)

donner l'automate déterministe **complet** minimal équivalent.

Dans votre réponse, les numéros/noms donnés aux états ne sont pas importants.

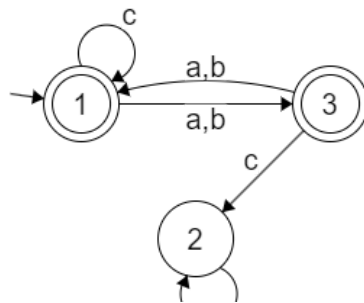
(votre réponse n'est pas évaluée durant le contrôle, donc toute réponse allume vert)

**Réponse :** (régime de pénalités : 0 %)

Réinitialiser la réponse

Help

Clear



	Test	Résultat attendu	Résultat obtenu	
✓	dfa_s.is_complete()	True	True	✓
✓	dfa_s.c.states	[0, 1, 2]	[0, 1, 2]	✓
✓	dfa_s.c.accept_states	[0, 1]	[0, 1]	✓
✓	trans(0)	(1, 1, 0)	(1, 1, 0)	✓
✓	trans(1)	(0, 0, 2)	(0, 0, 2)	✓
✓	trans(2)	(2, 2, 2)	(2, 2, 2)	✓

Tous les tests ont été réussis ! ✓

Il s'agit de minimiser le DFA donné.

Piège à éviter: l'automate n'est pas complet. Il faut commencer par le compléter. On ajoute un nouvel état poubelle  $q_5$ .

Quand on est dans l'état  $q_1$  et  $q_3$  et qu'on lit la lettre  $c$ , on transitionne dans l'état poubelle.

Première étape de l'algorithme, on a une partition en deux parties:  $\{q_0, q_1, q_2, q_3\}$  (les états acceptants) d'un côté, et  $\{q_5\}$  (état non acceptant) de l'autre.

On se rend compte qu'il faut séparer  $\{q_0, q_2\}$  et  $\{q_1, q_3\}$  car quand on lit la lettre  $c$  dans le premier cas on reste dans  $\{q_0, q_1, q_2, q_3\}$  et dans l'autre on va en  $\{q_5\}$ .

Deuxième étape de l'algorithme,, on a une partition en trois parties:  $\{q_0, q_2\}$ ,  $\{q_1, q_3\}$  et  $\{q_5\}$ .

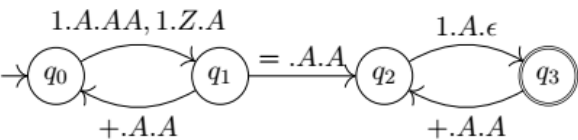
Pas d'incohérence cette fois. Donc on fusionne chaque sous-ensemble en un seul état.

► **Montrer / masquer la solution de l'auteur de la question (Python3)**

Correct

Note pour cet envoi : 2,00/2,00.

Considérons l'automate à pile suivant avec **Z** comme symbole de fond de pile:



Remplir le tableau suivant qui décrit l'exécution de cet automate à pile sur le mot  $1 + 1 = 1 + 1$ :

Attention: dans la pile, le sommet de la pile est à gauche. Mettre **e** à la place de epsilon pour indiquer la pile vide.

Ne pas laisser d'espace dans vos réponses.

État	Mot entrée	Pile
q0 ✓	1 + 1 = 1 + 1	Z ✓
q1 ✓	+ 1 = 1 + 1	A ✓
q0 ✓	1 = 1 + 1	A ✓
q1 ✓	= 1 + 1	AA ✓
q2 ✓	1 + 1	AA ✓
q3 ✓	+ 1	A ✓
q2 ✓	1	A ✓
q3 ✓	e	e ✓

Est-ce que le mot est accepté par l'automate à pile?

Oui ✓ .

## Question 12

Incorrect

Note de 0,00 sur 1,00

Considérons la grammaire:

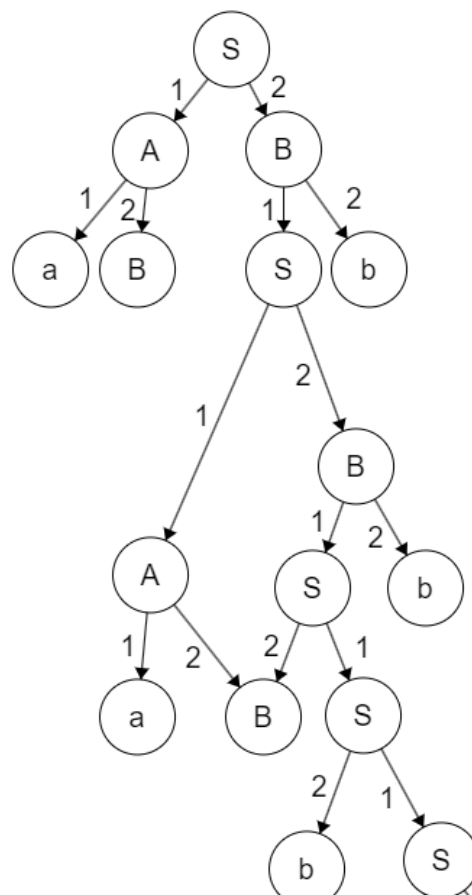
 $S \rightarrow AB \mid \epsilon$  $A \rightarrow aB$  $B \rightarrow Sb$ Trouver un arbre de dérivation pour le mot  $aabbbb$ .

Si un sommet a plusieurs enfants, mettre des numéros 1,2,3,4,... sur les arcs pour indiquer qui est le premier sommet, le second, ... Ne rien mettre dans un sommet à la place de  $\epsilon$ .

Réponse : (régime de pénalités : 0 %)

Réinitialiser la réponse

Help Clear



	Test	Résultat attendu	Résultat obtenu	
✖	len(arbres)	1	<pre>***Run error*** Traceback (most recent call last):   File "__tester__.python3", line 17, in &lt;module&gt;     arbres = moodle_to_forest(s_a)   File "arbre.py", line 54, in moodle_to_forest     raise Exception('Un sommet a plusieurs voisins entrants')\t\t\t\t Exception: Un sommet a plusieurs voisins entrants</pre>	✖

Le test a été interrompu à cause d'une erreur.

Montrer les différences

► **Montrer / masquer la solution de l'auteur de la question (Python3)**

Incorrect

Note pour cet envoi : 0,00/1,00.

**Question 13**

Correct

Note de 1,00 sur 1,00

Considérons la grammaire hors-contexte:

 $S \rightarrow AB \mid \epsilon$ 
 $A \rightarrow aB$ 
 $B \rightarrow Sb$ 

Dessiner un automate à pile avec un seul état qui reconnaît le même langage que la grammaire ci-dessus.

**Le symbole sur la pile à l'état initial est S.**

Vous pouvez écrire vos transitions au format **a.A.w** où

- **a** est une unique lettre de l'alphabet d'entrée (ou rien pour epsilon),
- **A** est une unique lettre de l'alphabet de la pile (ou rien pour epsilon), et
- **w** est un mot sur l'alphabet de la pile. **ATTENTION: La lettre la plus à gauche de w devient le sommet de la pile.**

Vous pouvez écrire toutes vos transitions sur plusieurs flèches où sur une seule flèches séparés par des virgules.

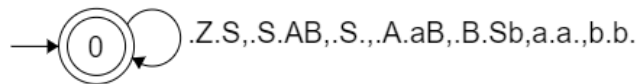
Exemples: **a.A.BA,b.B.BA**

**Réponse :** (régime de pénalités : 0 %)

Réinitialiser la réponse

Help

Clear



	Test	Got	Score	Total	
✖	nb d'état = 1 mots de lg 0 mots de lg 1 mots de lg 2 mots de lg 3 mots de lg 4 mots de lg 5 mots de lg 6 mots de lg 7 mots de lg 8 mots de lg 9	***Time limit exceeded***	Unknown field 'comment'	0.00	✖

► Montrer / masquer la solution de l'auteur de la question (Python3)

Correct

Note pour cet envoi : 0,00/1,00.

Commentaire :

Question 14

Correct

Note de 1,00 sur 1,00

Considérons la grammaire:

$S \rightarrow A \mid B$

$A \rightarrow B \mid A + B$

$B \rightarrow c \mid B * c$

Prouver que cette grammaire est ambiguë en dessinant deux arbres de dérivation différents pour un même mots accepté par cette grammaire.  
(vous devez trouver le mot vous-même)

*Si un sommet a plusieurs enfants, mettre des numéros 1,2,3,4,... sur les arcs pour indiquer qui est le premier enfant, le deuxième... Ne rien mettre dans un sommet à la place de  $\epsilon$ .*

Réponse : (régime de pénalités : 0 %)

Réinitialiser la réponse

HelpClear

S

A

B

c

S

B

c

	Test	Résultat attendu	Résultat obtenu	
✓	len(arbres)	2	2	✓
✓	"".join(feuilles(a1)) == "".join(feuilles(a2))	True	True	✓
✓	respecte_regles(a1,regles,True)	True	True	✓



	Test	Résultat attendu	Résultat obtenu	
✓	respecte_regles(a2,regles,True)	True	True	✓
✓	arbre_differeents(a1,a2)	True	True	✓

Tous les tests ont été réussis ! ✓

► **Montrer / masquer la solution de l'auteur de la question (Python3)**

Correct

Note pour cet envoi : 1,00/1,00.

Question 15

Correct

Note de 1,00 sur 1,00

Considérons la grammaire:  $S \rightarrow aaS \mid SbS \mid \epsilon$ .

Prouver que cette grammaire est ambiguë en dessinant deux arbres de dérivation différents pour un même mots accepté par cette grammaire.

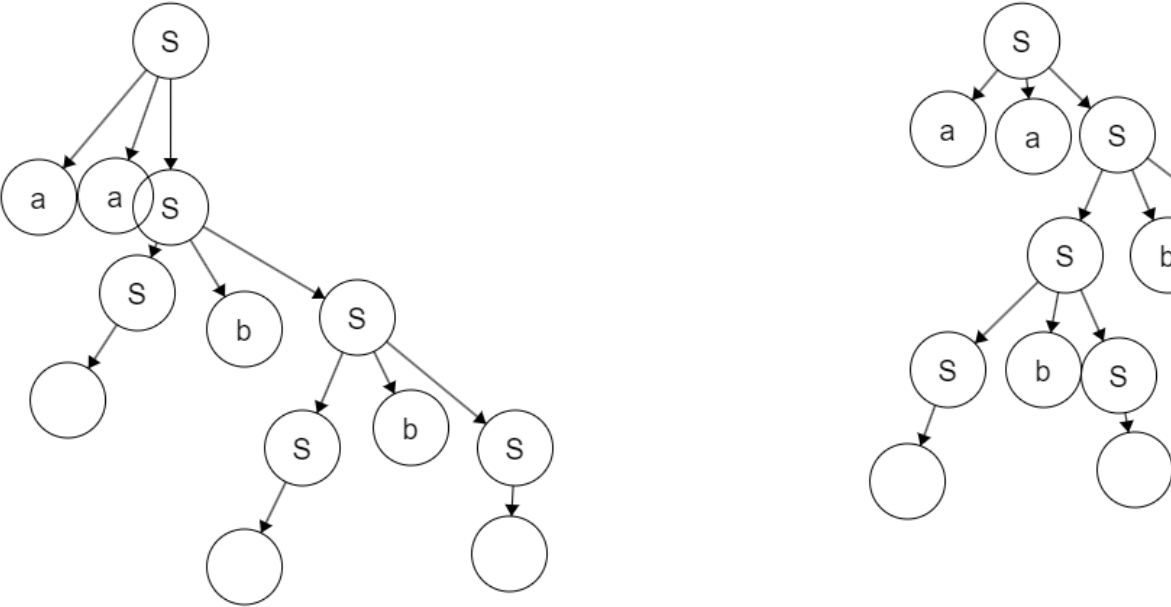
(vous devez trouver le mot vous-même)

*Si un sommet a plusieurs enfants, mettre des numéros 1,2,3,4,... sur les arcs pour indiquer qui est le premier enfant, le deuxième... Ne rien mettre dans un sommet à la place de  $\epsilon$ .*

Réponse : (régime de pénalités : 0 %)

Réinitialiser la réponse

HelpClear



	Test	Résultat attendu	Résultat obtenu	
✓	len(arbres)	2	2	✓
✓	"".join(feuilles(a1)) == "".join(feuilles(a2))	True	True	✓
✓	respecte_regles(a1,regles,True)	True	True	✓
✓	respecte_regles(a2,regles,True)	True	True	✓
✓	arbre_differeents(a1,a2)	True	True	✓

Tous les tests ont été réussis ! ✓

► **Montrer / masquer la solution de l'auteur de la question (Python3)**

Correct

Note pour cet envoi : 1,00/1,00.

Question 16

Correct

Note de 2,00 sur 2,00

On considère le langage  $L = \{a^n b^n c^m d^m \mid n, m \geq 1\}$ .  
(Donc  $\epsilon, ab, cd \notin L$ )  
Dessiner un automate à pile qui reconnaît le langage  $L$ .

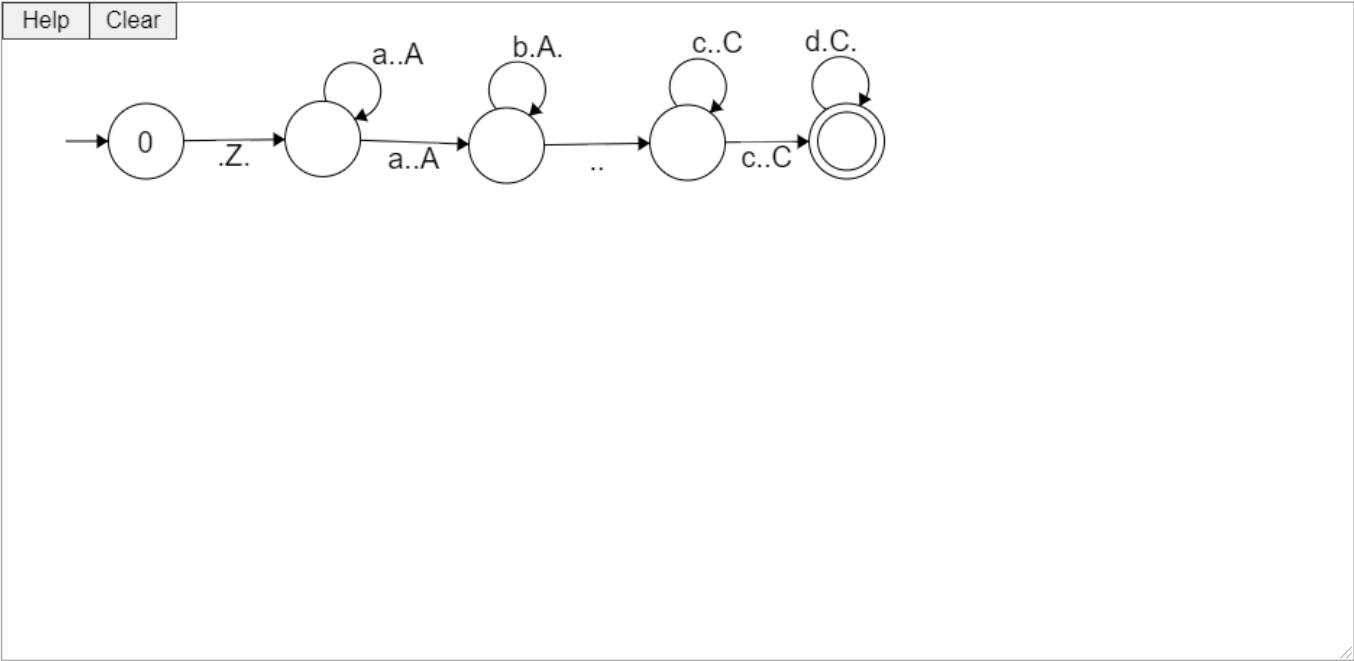
Le symbole sur la pile à l'état initial est **Z**.  
Vous pouvez écrire vos transitions au format **a.A.w** où

- a** est une unique lettre de l'alphabet d'entrée (ou rien pour epsilon),
- A** est une unique lettre de l'alphabet de la pile (ou rien pour epsilon), et
- w** est un mot sur l'alphabet de la pile. **ATTENTION: La lettre la plus à gauche de w devient le sommet de la pile.**

Vous pouvez écrire toutes vos transitions sur plusieurs flèches où sur une seule flèches séparés par des virgules.  
Exemples: **a.A.BA,b.B.BA**

Réponse : (régime de pénalités : 0 %)

Réinitialiser la réponse



	Test	Got	Expected	Score	Total	
✓	mots de lg 0	set()	set()	0.1	1.00	✓
	mots de lg 1	set()	set()	0.1		
	mots de lg 2	set()	set()	0.1		
	mots de lg 3	set()	set()	0.1		
	mots de lg 4	{'abcd'}	{'abcd'}	1.1		
	mots de lg 5	set()	set()	0.1		
	mots de lg 6	{'aabbcd', 'abccdd'}	{'aabbcd', 'abccdd'}	2.1		
	mots de lg 7	set()	set()	0.1		
	mots de lg 8	{'aabbccdd', 'abccddd', 'aaabbbcd'}	{'aabbccdd', 'abccddd', 'aaabbbcd'}	3.1		

Tous les tests ont été réussis ! ✓

► Montrer / masquer la solution de l'auteur de la question (Python3)

Correct

Note pour cet envoi : 2,00/2,00.



**Question 17**

Partiellement correct

Note de 1,23 sur 2,00

On considère le langage  $L = \{w \in \{a, b\}^* \mid |w|_a < |w|_b\}$  des mots sur l'alphabet  $\{a, b\}$  avec strictement moins de  $a$  que de  $b$ .

Dessiner un automate à pile qui reconnaît le langage  $L$ .

Le symbole sur la pile à l'état initial est **Z**.

Vous pouvez écrire vos transitions au format **a.A.w** où

- **a** est une unique lettre de l'alphabet d'entrée (ou rien pour epsilon),
- **A** est une unique lettre de l'alphabet de la pile (ou rien pour epsilon), et
- **w** est un mot sur l'alphabet de la pile. **ATTENTION: La lettre la plus à gauche de w devient le sommet de la pile.**

Vous pouvez écrire toutes vos transitions sur plusieurs flèches où sur une seule flèches séparés par des virgules.

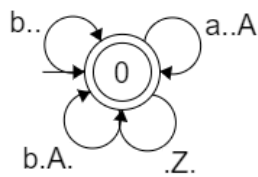
Exemples: **a.A.BA,b.B.BA**

**Réponse :** (régime de pénalités : 0 %)

Réinitialiser la réponse

Help

Clear



	Test	Got	Expected	Score	Total	
✓	mots de lg 0 mots de lg 1 mots de lg 2 mots de lg 3 mots de lg 4 mots de lg 5 mots de lg 6 mots de lg 7	{''} {'b'} {'bb', 'ab'} {'bab', 'bbb', 'abb'} {'abbb', 'aabb', 'babb', 'abab', 'bbbb', 'bbab'} 10 20 35	set() {'b'} {'bb'} {'bab', 'bbb', 'bba', 'abb'} {'abbb', 'babb', 'bbbb', 'bbab', 'bbba'} 16 22 64	0.1 1.1 1.1 3.1 4.1 10.1 15.1 35.1	0.61	✓

► Montrer / masquer la solution de l'auteur de la question (Python3)

Partiellement correct

Note pour cet envoi : 1,23/2,00.

### Question 18

Partiellement correct

Note de 0,60 sur 1,00

L et M étant 2 langages, pour chacun des langages ci-dessous, dire si il est :

- (dans tous les cas) hors-contexte
- (dans tous les cas) non hors-contexte
- parfois hors-contexte parfois non hors-contexte (ça dépend de L et de M)
- aucune des 3 propositions précédentes n'est vraie.

(ci-dessous, on note hc pour hors-contexte)

Si L est hc alors  $\{u.u \mid u \in L\}$  est

non hc



Si L et M sont hc alors  $L \cup M$  est

hc



Si L et M sont hc alors  $L.M = \{u.v \mid u \in L \text{ et } v \in M\}$  est

non hc



Si L est un langage hc et M un langage tel que  $M \subseteq L$  alors M est

parfois hc parfois non hc (ça dépend de L et de M)



Si L et M sont hc alors  $L \cap M$  est

parfois hc parfois non hc (ça dépend de L et de M)



Votre réponse est partiellement correcte.

Vous en avez sélectionné correctement 3.

La classe (= l'ensemble) des langages hors-contexte est close pour les opérations ensemblistes suivantes :

- union
- concaténation
- \*

Mais la classe (= l'ensemble) des langages hors-contexte n'est pas close par intersection, ni par inclusion.

La réponse correcte est :

Si L est hc alors  $\{u.u \mid u \in L\}$  est  $\rightarrow$  parfois hc parfois non hc (ça dépend de L et de M),

Si L et M sont hc alors  $L \cup M$  est  $\rightarrow$  hc,

Si L et M sont hc alors  $L.M = \{u.v \mid u \in L \text{ et } v \in M\}$  est  $\rightarrow$  hc,

Si L est un langage hc et M un langage tel que  $M \subseteq L$  alors M est  $\rightarrow$  parfois hc parfois non hc (ça dépend de L et de M),

Si L et M sont hc alors  $L \cap M$  est  $\rightarrow$  parfois hc parfois non hc (ça dépend de L et de M)

