```
;
==========================================================================
==========
; Ex2.1 (parité)

; Langage : les mots de {0,1}* avec un nombre pair de 1

; bon celui-là c'est juste un automate fini
; mettre q0 comme état initial

q0 0 0 r q0
q0 1 1 r q1
q0 _ _ * halt-accepte

q1 0 0 r q1
q1 1 1 r q0
q1 _ _ * halt-rejette

;
==========================================================================
==========
; Ex2.2 (modulo4) format : 6 s'écrit 110

; Langage : les mots de {0,1}* codant en binaire un entier divisible
par 4

; on va a droite jusqu'à _
q0 0 0 r q0
q0 1 1 r q0
q0 _ _ l q1

; on teste les deux derniers digits
q1 0 0 l q2
q1 1 1 * halt-rejette
q1 _ _ * halt-rejette

q2 0 0 * halt-accepte
q2 1 1 * halt-rejette
q2 _ _ * halt-accepte

;
==========================================================================
==========
; Ex2.3 (prefixe) format: x#y ex: aba#ababa

; Langage : les mots de {a,b,#}* sous la forme x#y tels que x est un
préfixe de y

; dans un premier temps, on vérifie que l'input est sous la forme
x#y
; (et pas 00#10#1 par exemple, qui est dans {a,b,#}* mais pas dans
le langage que l'on souhaite reconnaître)

; on fait ça en une passe, et on reviens se placer en début d'input
```

```
q0 a a r q0
q0 b b r q0
q0 # # r q1
q0 _ _ * halt-rejette

q1 a a r q1
q1 b b r q1
q1 # # * halt-rejette
q1 _ _ l q2

q2 a a l q2
q2 b b l q2
q2 # # l q2
q2 _ _ r p0

; idée: on remplace les lettres de x par des blank (_) et celles de
y par des #, on garde l'info dans l'état quand on se déplace
; après avoir testé les 2 premiers lettres, on est donc dans l'état
0 et le ruban vaut __a###aba par exemple

; on efface la première lettre de x et on va dans l'état approprié
p0 a _ r p1a
p0 b _ r p1b
p0 # # * halt-accepte ; car le mot vide est préfixe de tout

; on se déplace jusqu'au premier #
p1a a a r p1a
p1a b b r p1a
p1a # # r p2a

p1b a a r p1b
p1b b b r p1b
p1b # # r p2b

; on se décale après les # et on teste la première lettre de y
; si la lettre était correcte, on reviens au début de l'input avec
q2
p2a a # l q2
p2a b b * halt-rejette
p2a # # r p2a
p2a _ _ * halt-rejette

p2b a a * halt-rejette
p2b b # l q2
p2b # # r p2b
p2b _ _ * halt-rejette

;
==========================================================================
==========
; Ex2.3 bis (autre solution)
; pour pouvoir se servir de la machine reconnaissant préfixe dans la
suite,
; on en fait une version alternative, qui n'efface pas d'information
```

```
; (on utilise A et B pour remplacer les a et b de x et y au fur et à mesure du calcul)

; dans un premier temps, on vérifie que l'input est sous la forme x#y

; on fait ça en une passe, et on reviens se placer en début d'input
q0 a a r q0
q0 b b r q0
q0 # # r q1
q0 _ _ * halt-rejette

q1 a a r q1
q1 b b r q1
q1 # # * halt-rejette
q1 _ _ l q2

q2 a a l q2
q2 b b l q2
q2 # # l q2
q2 A A r p0
q2 B B r p0
q2 _ _ r p0

; on reprends la machine prefixe, mais on ne perd pas d'information dans le calcul (on utilise A et B pour remplacer a et b)
; après avoir testé les 2 premiers lettres, on est donc dans l'état 0 et le ruban vaut ABa#ABaba par exemple
; on efface la première lettre de x et on va dans l'état approprié
p0 a A r p1a
p0 b B r p1b
p0 # # * halt-accepte ; car le mot vide est préfixe de tout

; on se déplace jusqu'au #
p1a a a r p1a
p1a b b r p1a
p1a # # r p2a

p1b a a r p1b
p1b b b r p1b
p1b # # r p2b

; on teste la première lettre de y
; si la lettre était correcte, on reviens au début de l'input avec p3
p2a a A l p3
p2a b b * halt-rejette
p2a A A r p2a
p2a B B r p2a
p2a _ _ * halt-rejette

p2b a a * halt-rejette
p2b b B l p3
p2b A A r p2b
```

```
p2b B B r p2b
p2b _ _ * halt-rejette

; on passe sur les A et B de y, puis une fois les # atteints
; on reviens avec q2 sur la première lettre non A ou B de x
p3 A A l p3
p3 B B l p3
p3 # # l q2

;
==========
; Ex2.4 (sous-mot) format: aba#aababab

; Langage : les mots de {a,b,#}* sous la forme x#y tels que x est un
facteur de y

; on va utiliser la machine préfixe (deuxième version),
; et on teste toutes les positions initiales possibles

; si le test de préfixe rejette, on efface la première lettre de y
(avec un #),
; on remplace les A et B par des a et b, et on relance le test de
préfixe sur x##y'

; dans un premier temps, on vérifie que l'input est sous la forme
x#y
; on fait ça en une passe, et on reviens se placer en début d'input

q0 a a r q0
q0 b b r q0
q0 # # r q1
q0 _ _ * halt-rejette

q1 a a r q1
q1 b b r q1
q1 # # * halt-rejette
q1 _ _ l q2

q2 a a l q2
q2 b b l q2
q2 # # l q2
q2 A A r p0
q2 B B r p0
q2 _ _ r p0

; puis on reprend préfixe telle qu'elle, mais on change  l'état
d’arrêt par un état préfixe-rejette

p0 a A r p1a
p0 b B r p1b
p0 # # * halt-accepte

p1a a a r p1a
```

```
p1a b b r p1a
p1a # # r p2a

p1b a a r p1b
p1b b b r p1b
p1b # # r p2b

p2a a A l p3
p2a b b * prefix-rejette
p2a # # r p2a ; on ajoute cette règle pour ignorer plusieurs #
conséutifs
p2a A A r p2a
p2a B B r p2a
p2a _ _ * prefix-rejette

p2b a a * prefix-rejette
p2b b B l p3
p2b # # r p2b ; on ajoute cette règle pour ignorer plusieurs #
conséutifs
p2b A A r p2b
p2b B B r p2b
p2b _ _ * prefix-rejette

p3 A A l p3
p3 B B l p3
p3 # # l q2

; on ne rejette que lorsque la tête de lecture est dans y, donc
depuis prefix-rejette,
; on se décale à gauche jusqu'au # en changeant les A et a et B en b
prefix-rejette a a l prefix-rejette
prefix-rejette b b l prefix-rejette
prefix-rejette A a l prefix-rejette
prefix-rejette B b l prefix-rejette
prefix-rejette # # r f1
prefix-rejette _ _ l prefix-rejette

; on efface la première lettre de y
f1 a # l f2
f1 b # l f2
f1 _ _ * halt-rejette ; si on a épuisé toutes les positions
possibles dans y, on rejette

; puis on reviens en position initiale en réinitialisant le ruban
sur x
f2 a a l f2
f2 b b l f2
f2 # # l f2
f2 A a l f2
f2 B b l f2
f2 _ _ r p0 ; on relance le teste de préfixe

;
==========
```

```
==========
; Ex3

0 * * * 0

;
================================================================================
==========
; Ex4.1 (sous-mot) x#y

; Machine variant -> Non-déterministic
; attention, le simulateur tire au sort un chemin au lieu de simuler
tout l'arbre, ainsi,
; tester sur "aba#babab", marche une fois sur 4

; Langage : les mots de {a,b,#}* sous la forme x#y tels que x est un
facteur de y

; dans un premier temps, on vérifie que l'input est sous la forme
x#y
; on fait ça en une passe, et on reviens se placer en début d'input
q0 a a r q0
q0 b b r q0
q0 # # r q1
q0 _ _ * halt-rejette

q1 a a r q1
q1 b b r q1
q1 # # * halt-rejette
q1 _ _ l q2

q2 a a l q2
q2 b b l q2
q2 # # l q2
q2 _ _ r n0

; de manière non-déterministe, on efface un prefixe de y : x####y'

n0 a a r n0
n0 b b r n0
n0 # # r n1

n1 a a l n2
n1 a # r n1
n1 b b l n2
n1 b # r n1
n1 _ _ l n2

n2 a a l n2
n2 b b l n2
n2 # # l n2
n2 _ _ r p0

; on vérifie en appelant la machine prefixe (première version) sur
```

```
x###y'

; on efface la première lettre de x et on va dans l'état approprié
p0 a _ r p1a
p0 b _ r p1b
p0 # # * halt-accepte ; car le mot vide est préfixe de tout

; on se déplace jusqu'au premier #
p1a a a r p1a
p1a b b r p1a
p1a # # r p2a

p1b a a r p1b
p1b b b r p1b
p1b # # r p2b

; on se décale après les # et on teste la première lettre de y
; si la lettre était correcte, on reviens au début de l'input avec
q2
p2a a # l q2bis
p2a b b * halt-rejette
p2a # # r p2a
p2a _ _ * halt-rejette

p2b a a * halt-rejette
p2b b # l q2bis
p2b # # r p2b
p2b _ _ * halt-rejette

q2bis a a l q2bis
q2bis b b l q2bis
q2bis # # l q2bis
q2bis _ _ r p0

;
==========
; Ex 4.2 (Sous-palindrome4) , tester sur babbaba, une chance sur 16
d'accepter

; On devine le point de départ

q0 a a r q0
q0 b b r q0
q0 a a * q0choice
q0 b b * q0choice
q0 _ _ * halt-rejette

; On devine quel palindrome

q0choice * * * q1
q0choice * * * q2
q0choice * * * q3
q0choice * * * q4
```

```
; On teste aaaa
q1 a a r 1.1
q1 b b * halt-rejette
q1 _ _ * halt-rejette

1.1 a a r 1.2
1.1 b b * halt-rejette
1.1 _ _ * halt-rejette

1.2 a a r 1.3
1.2 b b * halt-rejette
1.2 _ _ * halt-rejette

1.3 a a * halt-accepte
1.3 b b * halt-rejette
1.3 _ _ * halt-rejette

; On teste abba
q2 a a r 2.1
q2 b b * halt-rejette
q2 _ _ * halt-rejette

2.1 b b r 2.2
2.1 a a * halt-rejette
2.1 _ _ * halt-rejette

2.2 b b r 2.3
2.2 a a * halt-rejette
2.2 _ _ * halt-rejette

2.3 a a * halt-accepte
2.3 b b * halt-rejette
2.3 _ _ * halt-rejette

; On teste baab
q3 b b r 3.1
q3 a a * halt-rejette
q3 _ _ * halt-rejette

3.1 a a r 3.2
3.1 b b * halt-rejette
3.1 _ _ * halt-rejette

3.2 a a r 3.3
3.2 b b * halt-rejette
3.2 _ _ * halt-rejette

3.3 b b * halt-accepte
3.3 a a * halt-rejette
3.3 _ _ * halt-rejette

; On teste bbbb
q4 b b r 4.1
```

```
q4 a a * halt-rejette
q4 _ _ * halt-rejette

4.1 b b r 4.2
4.1 a a * halt-rejette
4.1 _ _ * halt-rejette

4.2 b b r 4.3
4.2 a a * halt-rejette
4.2 _ _ * halt-rejette

4.3 b b * halt-accepte
4.3 a a * halt-rejette
4.3 _ _ * halt-rejette

;
==========
; Ex5.1 (x->8x+3)

q0 0 0 r q0
q0 1 1 r q0
q0 _ 0 r q1

q1 _ 1 r q2

q2 _ 1 * halt

;
==========
; Ex5.2 ((x,y)->x+y) au format x#y avec x et y en binaire

; idée: pour faire x + y , on prends la dernière lettre de y, on
l'ajoute à x (avec propagation de retenue)
; puis on remplace la dernière lettre de x (0 par a, 1 par b),
; et on recommence (en ignorant les a et b lors du calcul)
; lorsque l'on a fini (y vide), on remplace les a et b par 0 et 1

q0 0 0 r q0
q0 1 1 r q0
q0 a a r q0
q0 b b r q0
q0 # # r q0
q0 _ _ l q1

q1 0 _ l q2.0
q1 1 _ l q2.1
q1 # _ l qdone

q2.0 0 0 l q2.0
q2.0 1 1 l q2.0
q2.0 # # l q3.0
```

```
q2.1 0 0 l q2.1
q2.1 1 1 l q2.1
q2.1 # # l q3.1

q3.0 0 a r q0
q3.0 1 b r q0
q3.0 a a l q3.0
q3.0 b b l q3.0
q3.0 _ a r q0

q3.1 0 b r q0
q3.1 1 a l q4.1
q3.1 a a l q3.1
q3.1 b b l q3.1
q3.1 _ b r q0

q4.1 0 1 r q0
q4.1 1 0 l q4.1
q4.1 _ 1 r q0

qdone a 0 l qdone
qdone b 1 l qdone
qdone 0 0 * halt
qdone 1 1 * halt
qdone _ _ * halt

;
==========
; Ex 6 format d'entrée : un ensemble d'entiers S codé en binaire par
x1#x2#x3#x4

; Langage : les mots de {0,1,#}* sous la forme x1#x2#...#xn tels
qu'il existe
; une partition de l'ensemble S={x1,...xn} en T et S\T de somme
égale

; on teste le format de l'entrée (#x1#x2#x3#x4), et de manière non-
déterministe on
; remplace des # par des !
; on ajoute aussi un symbole $ en fin et un # ou ! au début
q0 0 0 r q1
q0 1 1 r q1
q0 # # * halt-rejette
q0 _ _ * halt-rejette

q1 0 0 r q1
q1 1 1 r q1
q1 # # r q0
q1 # ! r q0
q1 _ $ l q2

; on se replace au début
q2 0 0 l q2
```

```
q2 1 1 l q2
q2 # # l q2
q2 ! ! l q2
q2 _ # r i0
q2 _ ! r i0

; les ! indiquent les x_i sélectionnés pour être dans l'ensemble T

; idée de la suite :
; en utilisant une variant de la question 5.2, on calcule la somme
des x_i précédés d'un !,
; et on range le résultat après le $, suivi d'un £
; on calcule aussi la somme des x_i précédés d'un !, et on stocke le
résultat après le £
; on teste l'égalité de ces deux mots (comme préfixe mais condition
d'arrêt légèrement différente)

;
==========================================================================
==========
; Ex 7 (Busy Beaver)

; C'est le gagnant actuel du concours du busy beaver à 6 états !
; (cf https://fr.wikipedia.org/wiki/Castor_affair%C3%A9)
; Il a été démontré que cette machine s'arrête, en au moins 7.4 ∗
10^36534 étapes,
; et après avoir écrit au moins 3.5 ∗ 10^18267 symboles 1 sur le
ruban.

q0 _ 1 r q1
q0 1 1 l q4

q1 _ 1 r q2
q1 1 1 r q5

q2 1 _ r q1
q2 _ 1 l q3

q3 1 _ l q2
q3 _ 1 r q4

q4 1 _ r q3
q4 _ 1 l q0

q5 1 1 r q2
q5 _ _ ∗ halt

;
==========================================================================
==========
```