1. Une machine de Turing à un ruban peut être simulée par une machine RAM standard (dont chaque registre contient un entier de n bits).
2. L'ensemble des programmes WHILE est un ensemble dénombrable.
3. Il est possible de définir une machine de Turing qui, étant donné le texte d'un programme WHILE (avec ses entrées), effectue le même calcul que ce programme.
4. Il est possible d'écrire un programme (dans votre langage préféré) qui, étant donné la table de transition d'une machine de Turing $\mathcal{M}$ et une configuration initiale, renvoie VRAI si en partant de cette configuration initiale, la suite des configurations est finie et FAUX sinon.

1. FAUX : une machine RAM à registres bornés n'a qu'une mémoire bornée.

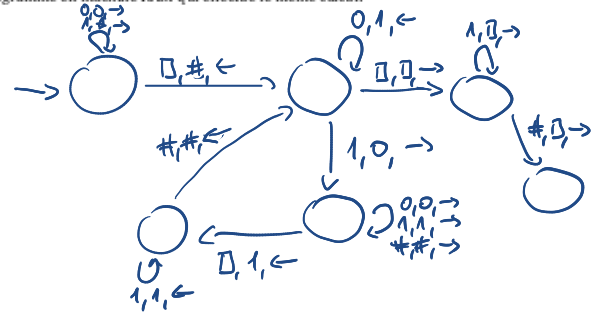
2. VRAI : un programme WHILE est une chaîne de caractère finie sur un alphabet fini.

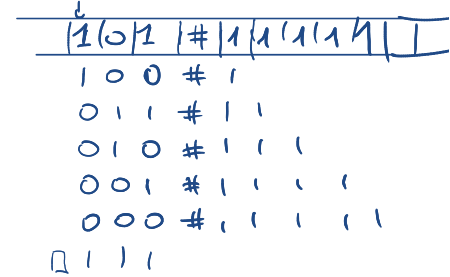
3. VRAI : on combine la machine universelle (qui exécute n'importe quelle machine de Turing) et la transformation algorithmique d'un programme WHILE en machine de Turing.

4. FAUX : c'est le problème de l'arrêt "croisé" : une formalisation des algos ne peut pas décider l'arrêt d'une autre formalisation

1. Décrire une machine de Turing qui, étant donné un entier n écrit en binaire sur son ruban, calcule le mot 1^n (n fois le symbole 1). Par exemple, sur l'entrée 101, la machine doit calculer 11111. *Remarque. La machine doit être donnée explicitement par un automate ou une table de transition. Il est conseillé, mais pas obligatoire, de n'utiliser qu'un seul ruban.*
2. En utilisant l'égalité $\binom{n}{2} = \sum_{i=0}^{n-1} i$, écrire un programme WHILE qui étant donné n (dans la variable x_1), calcule $\binom{n}{2}$ (dans la variable x_0). *Il est autorisé d'utiliser l'instruction d'addition $x_i \leftarrow x_j + x_k$.*
3. Traduire ce programme en machine RAM qui effectue le même calcul.

1.





2.
```
X1 ← X1 - 1
while X1 ≠ 0:
    X0 ← X0 + X1
    X1 ← X1 - 1
```

```
while X1 ≠ 0:
    X1 ← X1 - 1
    X0 ← X0 + X1
```

3.
1. JUMP(R_1, 8)
2. DEC(R_1)
3. COPY(R_2, R_1)
4. JUMP(R_2, 1)
5. DEC(R_2)
6. INC(R_0)
7. JUMP(R_3, 4)
8. STOP

$R_0 \leftarrow R_0 + R_1$

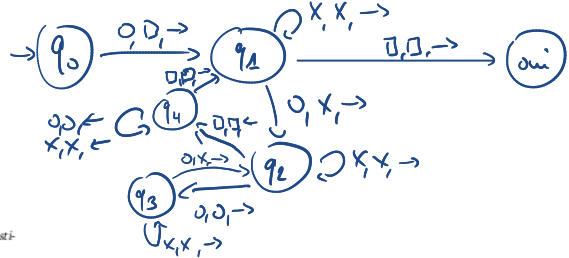
q_0	0	□	→	q_1
q_1	0	X	→	q_2
	X	X	→	q_1
	□	□	→	oui
q_2	0	0	→	q_3
	X	X	→	q_2
	□	□	←	q_4
q_3	0	X	→	q_2
	X	X	→	q_3
q_4	0	0	←	q_4
	X	X	←	q_4
	□	□	→	q_1

1. Parmi les mots 0, 00, 000 et 000000, lesquels appartiennent à $L(\mathcal{M})$? *Justifier en donnant la configuration finale atteinte.*

On souhaite maintenant comprendre ce que fait cette machine. Supposons que l'entrée est 0^n pour un certain $n > 1$. Après une étape de calcul, la machine arrive dans la configuration $q_i : \ddot{0}0^{n-1}$. *Note. La notation $\ddot{w}$ signifie que la tête est sur la première lettre de w.*
Au cours du calcul, certains 0 sont remplacés par des X. À tout moment du calcul, le mot inscrit sur le ruban est donc un mot $w \in \{0, X\}^*$.

2. Supposons que la machine est dans la configuration $q_i : \ddot{w}$ où w possède k symboles 0. On écrit $k = 2^t \cdot m - 1$ avec m impair, c'est-à-dire qu'on *factorise* la plus grande puissance de 2 possible dans $k+1$.
Exemples. Pour la configuration $q_i : \ddot{X}0X0$, $k = 2 = 2^0 \times 3 - 1$ donc $t = 0$ et $m = 3$; pour la configuration $q_i : \ddot{0}00$, $k = 3 = 2^2 \times 1 - 1$ donc $t = 2$ et $m = 1$.
   i. Montrer que si $t = 0$ et $m > 1$ (donc $k = m-1$ est pair), alors la machine se bloque en q_3.
   ii. Montrer que si $t > 0$ (donc $k = 2^t \cdot m - 1$ est impair), la machine finit par revenir dans une configuration $q_i : \ddot{w}'$ où w' possède $2^{t-1} \cdot m - 1$ symboles 0.
   iii. Que se passe-t-il si $k = 0$?

3. i. Si l'entrée sur le ruban est 0^n, déduire de ce qui précède la condition sur n pour que la machine accepte l'entrée.
   ii. Décrire aussi précisément que possible le langage $L(\mathcal{M})$.



1. $q_0 : \ddot{0} \vdash q_1 : \ddot{\square} \vdash \text{oui} : \ddot{0}$ ✓ $0 \in L(\mathcal{M})$

$q_0 : \ddot{0}0 \vdash q_1 : \ddot{0} \vdash q_2 : X\ddot{\square} \vdash q_4 : X\ddot{0} \vdash^* q_1 : \ddot{X} \vdash^* \text{oui} : X\ddot{\square}$ ✓

$q_0 : \ddot{0}00 \vdash q_1 : \ddot{0}0 \vdash q_2 : X\ddot{0} \vdash q_3 : X0\ddot{\square}$ ✗ $000 \notin L(\mathcal{M})$

$q_0 : \ddot{0}00000 \vdash^* q_3 : X0\ddot{0}00 \vdash q_2 : X0X\ddot{0}0 \vdash q_3 : X0X0\ddot{0}$

$\vdash q_2 : X0X0X\ddot{\square} \vdash^* q_1 : \ddot{X}0X0X \vdash^* q_2 : XX\ddot{X}0X$

$\vdash q_3 : XXX0\ddot{X} \vdash q_3 : XXX0X\ddot{\square}$ ✗ $000000 \notin L(\mathcal{M})$

2. i. Quand on passe de q_1 à q_2, on supprime un 0. Donc on arrive en q_2 avec $k-1$ zéros (et $k-1$ est impair). À chaque lecture d'un 0, on alterne entre q_2 et q_3. Donc après avoir lu tous les zéros, on est en q_{13}. On ignore les X en se déplaçant à droite donc on finit toujours en q_3 avec la lecture d'un □.

ii. Comme dans la question précédente, on alterne entre q_2 et q_3 mais comme k est impair, on a fini de lire tous les 0 en q_2. On lit un □, on passe en q_4, on rembobine au début du mot et on passe en q_2. On a effacé un 0 en passant de q_1 à q_2 (il en reste $k-1$) puis on en efface un sur deux en passant de q_3 à q_2 : au total, on efface $\frac{k-1}{2}+1$ zéros $= \frac{2^t m-2}{2}+1 = 2^{t-1}m$.
Donc il en reste $2^t m - 1 - 2^{t-1} m = 2^{t-1}m - 1$

iii. Si $k=0$, on passe dans l'état "oui".

3.i. Sur l'entrée 0^n, on passe en q_1 avec 0^{n-1}. Donc $k = n-1$ et on écrit $k+1 = 2^t \cdot m$.

→ Si $t=0$ et $m>1$, le mot est rejeté (q.2.i)

→ Si $t>0$, on revient en q_1 avec $2^{t-1} \cdot m + 1$ zéros (q2.ii) et on boucle tantque $t>0$: ⇝ on finit par revenir en q_1 avec $t=0$: il reste $m-1$ zéros

→ $m = 1$ ⇒ mot accepté (q.2.iii)

→ $m > 1$ ⇒ mot rejeté (q.2.i)

⇒ 0^n est accepté si $n = k+1 = 2^t \cdot m$ avec $m=1$ ⇝ $n = 2^t$

ii. $L(\mathcal{TL}) = \{ w \in \{0,x\}^* : w$ possède 2^t zéros pour $t \geq 0 \}$.
et qui commence par 0