On appelle automate un automate déterministe complet. Précisément, un automate est la donnée d'un quintuplet $A = \langle Q, \Sigma, \delta, F, q_0 \rangle$ où Q est un ensemble fini d'états, $F \subset Q$ est l'ensemble des états finals, $q_0 \in Q$ est l'état initial, Σ est un alphabet fini et δ est une application de $Q \times \Sigma \to Q$.

On étend la fonction δ à une fonction $\delta^* : Q \times \Sigma^* \to Q$ définie par récurrence par :

- $\forall q \in Q, \quad \delta^*(q, \varepsilon) = q$,
- $\forall a \in \Sigma, \forall u \in \Sigma^*, \forall q \in Q, \quad \delta^*(q, au) = \delta^*(\delta(q, a), u)$

Soit $q \in Q$ et $u \in \Sigma^*$, on notera de façon plus concise $q \cdot u$ pour $\delta^*(q, u)$.

Un mot $u \in \Sigma^*$ est accepté par l'automate A si et seulement s'il existe $\delta^*(q_0, u) \in F$. Le langage des mots acceptés par l'automate A noté $\mathscr{L}(A)$ est l'ensemble des mots acceptés par A.

Dans tout le TP l'alphabet Σ est égal à $\{0, 1\}$.

On implémente ici les automates déterministes complets sous Caml à l'aide du type enregistrement suivant :

```
type automate = {  etats : int ; delta : int array array ;
                   final : bool array ; initial : int } ;;
```

Soit `a` un tel automate,

- `a.etats` désigne le nombre d'états de l'automate. Les états seront les entiers naturels strictement inférieurs à `a.etats`.
- le tableau `a.delta` aura `a.etats` lignes et $2 = \mathrm{Card}(\Sigma)$ colonnes. Pour `q` compris entre 0 et `a.etats`-1 et `u` appartenant à $\{0, 1\}$ `a.delta.(q).(u)` désigne $q \cdot u$.
- le tableau de booléens `a.final` est défini de telle sorte que, pour `q` compris entre 0 et `a.etats`-1, `a.final.(q)` est égal à `true` si et seulement si `q` est un état final.
- l'entier `a.initial` est le numéro de l'état initial.

On utilisera le type `int` pour définir les éléments de $\Sigma = \{0, 1\}$. Les éléments de Σ^* seront vus comme des listes d'éléments de Σ. On utilisera donc

```
type mot = int list
```

1) Soit $x \in \mathbb{N}$. On considère la suite $(u_i)_{i \geq 0}$ définie par récurrence par

$$u_0 = x \qquad \text{et} \qquad u_{i+1} = 3321 u_i + 5701$$

Soit n un entier naturel non nul. On définit l'automate complet $A(x, n) = \langle \{0, \ldots, n-1\}, \{0, 1\}, \delta, F, 0 \rangle$ par

$$q \cdot a = u_{2q+a} \bmod n$$
$$q \in F \quad \text{ssi} \quad (u_q \mod n) \leq \frac{n}{3}$$

a) Écrire une fonction `suite : int -> int -> int -> int` telle que `suite x n k` renvoie la valeur de u_k modulo n où (u) est la suite définie ci-dessus en prenant $u_0 = x$.

b) Écrire une fonction `automateEx : int -> int -> automate` telle que `automateEx x n` renvoie l'automate $A(x, n)$.

c) (facultatif) Lors de l'appel de la fonction `automateEx` on calcule plusieurs fois les mêmes termes de la suite (u). Il est donc plus judicieux de stocker les termes de la suite (u) dans un tableau. Écrire une fonction `automateEx2 : int -> int -> automate` telle que `automateEx2 x n` renvoie l'automate $A(x, n)$ en utilisant un tableau pour stocker les valeurs de la suite (u).

d) Écrire une fonction `nbFinals : automate -> int` telle que `nbFinals a` renvoie le nombre d'états finals de l'automate `a`.

On vérifiera que pour les automates $A(5, 19)$, $A(5, 503)$, $A(16, 19)$ et $A(16, 503)$ la fonction renvoie $7, 168, 6$ et 168.

e) Soit $A = \langle Q, \Sigma, \delta, F, q_0 \rangle$ un automate.
On note $U_A = \{q \in Q \mid \exists a \in \{0,1\}, q \cdot a \in F\}$ et $V_A = \{q \in Q \mid \forall a \in \{0,1\}, q \cdot a \in F\}$

Écrire des fonction `cardU : automate -> int` et `cardV : automate -> int` telles que si `a` est un automate, `cardU a` et `cardV a` renvoie respectivement le cardinal des ensembles U et V définis ci-dessus.

On vérifiera qu'avec les mêmes automates que ci-dessus la fonction `cardU` renvoie 13, 281, 11 et 271 puis que `cardV` renvoie 3, 56, 2, 66.

2) a) Écrire une fonction `calcul : automate -> int -> mot -> int` telle que `calcul a q u` renvoie l'état $\delta^*(q, u)$.

b) En déduire une fonction `accepte : automate -> mot -> bool` telle que `accepte a u` renvoie un booléen qui vaut `true` si et seulement si u appartient au langage reconnu par A.

c) Écrire une fonction `nbMots : automate -> int -> int` telle que `nbMots a p` renvoie le nombre de mots de longueur p dans le langage $\mathscr{L}(A)$. On vérifiera qu'avec les mêmes automates qu'à la question 1.d la fonction `nbMots` renvoie 15, 8, 13 et 13 mots de longueur 5 et 436469, 353194, 355493 et 349726 mots de longueur 20.

3) On veut maintenant calculer le nombre de mots de longueur p accepté par un automate A. On utilise pour cela la programmation dynamique. Pour un automate A et un états $q \in Q$, on note $A[q]$ l'automate déduit de A en prenant q comme état initial. On note alors pour tout $i \in \mathbb{N}$, $H(q, i)$ le nombres de mots de longueur i dans le langage $\mathscr{L}(A[q])$.

a) Déterminer une formule de récurrence sur les termes $H(q, i)$

b) En déduire une fonction `nbMots2 : automate -> int -> int` telle que `nbMots2 a p` renvoie le nombre de mots de longueur p dans le langage $\mathscr{L}(A)$.

On pourra vérifier que la fonction donne les mêmes résultats que la précédente mais qu'elle s'exécute plus rapidement.