Exercice 1 :

Dire si les problèmes suivants sont décidables ou non (justifier) :

1. **Donnée:** Le code $\langle M \rangle$ d'une machine de Turing et un mot w. ;

   Question: Est-ce que la machine M boucle sur w ?

   On dit que M boucle sur w si, de la configuration initiale γ_0 on peut atteindre une configuration γ, de laquelle on peut à nouveau atteindre γ en au moins une étape : $\gamma_0 \vdash_M^* \gamma \vdash_M^+ \gamma$.

   Solution:
   On réduit le complémentaire du problème de l'arrêt sur le mot vide. f associe à $\langle M \rangle$ la machine M' suivante :
   - M' a pour alphabet $\Sigma' = \Sigma \uplus \{\beta\}$.
   - Pour chaque transition de M, M' va écraser le premier blanc à droite du ruban en remplaçant par β, puis simule la transition de M (β est traité comme un blanc pour cette transition).
   - Ensuite, si M est sur le point de s'arrêter, alors M' boucle.

   On montre que M' boucle ssi M s'arrête sur le mot vide.

2. **Donnée:** Le code d'une machine de Turing M ;

   Question: Il existe deux mots w_1, w_2 de même longueur tels que $w_1, w_2 \in L(M)$.

   Solution:
   C'est indécidable par le théorème de Rice. La propriété "pour tout n, $L(M)$ contient au plus un mot de longueur n" est en effet une propriété non triviale des langages récursivement énumérables.

Exercice 2 (Machines à registres/de Minsky) :

Une *machine à registres* est donnée par un ensemble fini d'états (contenant les états **accept** et **reject**), un état initial q_0, N *registres* $r_1, ..., r_N$ et une fonction de transition de $Q \times \{0,1\}^N$ dans $Q \times \{+1, 0, -1\}^N$: $\delta(q, \alpha_1, ..., \alpha_N) = (q', d_1, ..., d_N)$ telle que si $\alpha_I = 0$, alors $d_i \neq -1$.

Une *configuration* de la machine est donnée par N entiers en base 1 (les contenus de registres) et un état. Un mouvement de la machine est une relation entre configurations :

$$q, k_1, ..., k_N \vdash_M q', k'_1, ..., k'_N$$

ssi il existe une transition de la machine :

$$\delta(q, \alpha_1, ..., \alpha_N) = (q', d_1, ..., d_N)$$

telle que $\alpha_i = 0$ si $k_i = 0$ et $\alpha_i = 1$ sinon, et, pour tout i, $k'_i = k_i + d_i$.

Un *calcul* de la machine sur la donnée n_0 est une suite de configurations, la première étant la configuration initiale $(q_0, n_0, 0, ..., 0)$ et telle que la $i$ème configuration est obtenue par un mouvement de la machine sur la $i-1$ème configuration.

Une machine à registres *accepte* l'entier n_0, si le calcul de M sur n_0 s'arrête en **accept**.

1. Soit $k > 1$. Montrer qu'on peut construire une machine à registres M_k telle que, sur la donnée $n \in \mathbb{N}$, M_k s'arrête dans une configuration où le premier registre contient r et le deuxième registre contient q, où q et r sont respectivement le quotient et le reste de la division euclidienne de n par k.

Solution:
L'idée générale est la suivante :

- On commence par recopier le premier registre dans le second (en vidant le premier).
- Puis, pour k décréments du deuxième registre, on incrémente une fois le premier.
- Enfin, selon l'état d'arrivée, on inscrit le reste dans le deuxième registre.

Les états de la machine sont $\{q_i \mid 0 \leq i \leq k+1\} \cup \{q_i^r \mid 1 \leq i \leq k\}$, et les transitions sont les suivantes :

- $q_0, n_1, n_2 \to q_0, n_1 - 1, n_2 + 1$ si $n_1 > 0$
- $q_0, 0, n_2 \to q_1, 0, n_2$
- $q_i, n_1, n_2 \to q_{i+1}, n_1, n_2 - 1$ si $n_2 > 0$, pour $1 \leq i \leq k$
- $q_{k+1}, n_1, n_2 \to q_1, n_1 + 1, n_2$
- $q_i, n_1, 0 \to q_i^r, n_1, 0$, pour $1 \leq i \leq k$
- $q_i^r, n_1, n_2 \to q_{i-1}^r, n_1, n_2 + 1$, pour $i > 1$
- $q_1^r, n_1, n_2 \to \textbf{accept}, n_1, n_2$

2. Soit $\Sigma \setminus \{\$, B\} = \{a_1, ..., a_k\}$. Pour $w \in (\Sigma \setminus \{\$, B\})^*$, on note $c_k(w)$ l'entier dont l'écriture en base $k+1$ est w. Montrer que, si L est récursivement énumérable, il existe une machine à 4 registres qui accepte $\{c_k(w) \mid w \in L\}$.

Solution:
On note $\overline{w}$ le miroir d'un mot w. Si L est récursivement énumérable, alors L est accepté par une machine M. On considère alors une machine de Minsky M' à 4 registres qui simule M de la manière suivante : à une configuration w_1, q, w_2 de M correspond la configuration q, q_1, r_1, q_2, r_2 de M' où q_1, r_1 sont le quotient et le reste de la division euclidienne de $c_k(\overline{w_1})$ par $k+1$, et q_2, r_2 sont le quotient et le reste de la division euclidienne de $c_k(\overline{w_2})$ par $k+1$.
Pour chaque état q de M, M' a en réalité des états (q, q', q'') où $q', q'' \in \{\textbf{accept}\} \cup \{q_i \mid 0 \leq i \leq k+1\} \cup \{q_i^r \mid 1 \leq i \leq k\}$. Dans les étapes ci-dessous, il faut en fait choisir la "bonne" copie de q et q' selon quelle opération on s'apprête à faire sur les registres. Pour simplifier les explications qui suivent, on parle simplement de q et q'.
À une transition $q, a \to q', a', d$ on associe le calcul suivant dans M' :

- Si $d = \downarrow$:
  - on passe de q à q',
  - de $r_2 = a$ à $r_2 = a'$,
  - les autres registres ne changent pas.
- Si $d = \to$:
  - on passe de q à q',
  - de q_1, r_1 à $(k+1)q_1 + r_1, 0$ (l'inverse de la machine M_{k+1} de la question 1), puis à $(k+1)q_1 + r_1, a'$,
  - de $q_2, r_2 = a$ à $q_2, 0$ puis à q_2', r_2' quotient et reste de la division de q_2 par $k+1$.
- Si $d = \leftarrow$:
  - on passe de q à q',

- de $q_1, r_1, q_2, r_2 = a$ à q_1, r_1, q_2, a', puis à $q_1, r_1, (k+1)q_2 + a', 0$ (l'inverse de la machine M_{k+1} de la question 1), puis à $q_1, 0, (k+1)q_2 + a', r_1$,
- et enfin à $q_1', r_1', (k+1)q_2 + a', r_1$ où q_1', r_1' sont le quotient et le reste de la division de q_1 par $k+1$.

3. Montrer le même résultat que celui de la question précédente, mais avec une machine à 2 registres (au lieu de 4). [1]

Solution:

Soit N le code de (n_1, n_2, n_3, n_4) comme dans l'énoncé. Les états de la nouvelle machine à 2 registres sont $Q \times \{0, 1, U\}^4 \times (Q_2 \uplus Q_3 \uplus Q_5 \uplus Q_7 \uplus \{idle\}) \times \{-1, 0, +1\}^4 \times (\{idle\} \uplus (Q_2' \uplus Q_3' \uplus Q_5' \uplus Q_7'))$ où :

- Q est l'ensemble des états de la machine à 4 registres.
- $Q_i, Q_i', i = 2, 3, 5, 7$ sont les états de machines auxiliaires définies ci-après.
- $\{0, 1, U\}^4$ est le codage du statut de n_1, n_2, n_3, n_4 : 0 pour un registre nul, 1 pour un registre non nul, U pour un registre non encore testé.
- $\{-1, 0, +1\}^4$ est le codage des opérations restant à effectuer sur chaque registre.

Montrons d'abord comment tester la nullité des registres. On montre comment, à partir de $(q, s, idle, 0^4, idle), N, 0$ où s contient au moins un U, on atteint $(q, s', idle, 0^4, idle), N, 0$ dans lequel s' contient un U de moins que s. Soit i l'indice de la plus petite composante de s qui vaut U. On effectue alors successivement :

- la division de N par i en utilisant la machine M_i de la question 1. Le quotient de N est dans r_1, le reste est dans r_2. Les états utilisés dans ce calcul sont ceux de Q_i, toutes les autres composantes de l'état sont inchangées.
- Si $r_2 = 0$ (resp. $r_2 \neq 0$) on passe dans l'état (q, s', e) tel que s' est obtenu en remplaçant le U de la $i$ème composante par 1 (resp. 0). e est l'état initial de la machine M_i'.
- On calcule $i \times r_1 + r_2$, le résultat étant dans r_1 : ceci permet de revenir à la configuration $(q, s', idle), N, 0$.

En itérant 4 fois les étapes ci-dessus, on passe d'une configuration $(q, U^4, idle, 0^4, idle), N, 0$ à une configuration $(q, s, idle, 0^4, idle), N, 0$ où s rend compte du test à zéro de chacun des 4 registres. Si alors $q, s \to q', d_2, d_3, d_5, d_7$, on passe dans la configuration $(q', U^4, idle, (d_2, d_3, d_5, d_7), idle), N, 0$. De même que ci-dessus pour les tests à zéro, on effectue maintenant chacune des opérations sur les registres : si $(d_2, d_3, d_5, d_7) \neq (0, 0, 0, 0)$, soit i le plus petit indice tel que $d_i \neq 0$. Si $d_i = -1$, on effectue alors la division de N par i (en utilisant une copie de M_i et les états Q_i') et on met d_i à 0 une fois l'opération effectuée. De même, si $d_i = +1$, on effectue la multiplication de N par i et on met à jour d_i à 0.

À l'issue de ces étapes, on est passé d'une configuration $(q', U^4, idle, (d_2, d_3, d_5, d_7), idle), N, 0$ à une configuration $(q', U^4, idle, 0^4, idle), N', 0$ qui code la configuration suivante de la machine à 4 registres.

Ceci montre que, si L est le langage accepté par une machine à 4 registres, il existe une machine à 2 registres qui accepte L.

4. Montrer que le problème de savoir si une machine à 2 registres accepte au moins un entier est indécidable.

1. Indication : on pourra considérer le codage de 4 entiers en un seul par $\overline{(n_1, n_2, n_3, n_4)} = 2^{n_1} \times 3^{n_2} \times 5^{n_3} \times 7^{n_4}$.

Solution:
On construit alors à partir de l'entrée w d'une machine de Turing M l'entier $c_k(\overline{w})$.
On réduit alors le problème du vide des machines de Turing à celui du vide des machines à 2 compteurs.

Exercice 3 (Théorème de Rice-Shapiro) :
Soit $\mathcal{P}$ une propriété des langages récursivement énumérables. ($\mathcal{P}$ est un ensemble de codes de machines de Turing $\langle M \rangle$ tel que, si $L(M) = L(M')$ et $\langle M \rangle \in \mathcal{P}$, alors $\langle M' \rangle \in \mathcal{P}$. Par abus de langage, on dira alors que $L \in \mathcal{P}$ s'il existe une machine de Turing M telle que $L(M) = L$ et $\langle M \rangle \in \mathcal{P}$).

1. Montrer que, s'il existe deux langages récursivement énumérables $L_1 \subseteq L_2$ tels que $L_1 \in \mathcal{P}$ et $L_2 \notin \mathcal{P}$, alors $\mathcal{P}$ n'est pas récursivement énumérable.

Solution:
On réduit le problème du non-arrêt de M sur w (qui n'est pas récursivement énumérable). Soit M_1 et M_2 des machines qui, respectivement, acceptent L_1 et L_2 tels que $L_1 \subseteq L_2$, $L_1 \in \mathcal{P}$, $L_2 \notin \mathcal{P}$. On peut supposer sans perte de généralité que M_1 ne s'arrête jamais en **reject**. On construit alors la machine M' qui, sur la donnée x effectue les opérations suivantes :

Algorithm 1: M'

```
1  for i = 0 to +∞ do
2  |  Simuler i étapes de calcul de M sur w
3  |  if M ne s'arrête pas en moins de i étapes then
4  |  |  Simuler i étapes de calcul de M_1 sur x
5  |  |  if M_1 accepte x en moins de i étapes then
6  |  |  |  accept
7  |  |  end
8  |  else
9  |  |  simuler M_2 sur x
10 |  end
11 end
```

- Si M ne s'arrête pas sur w, alors, pour tout i, on exécute la ligne 4. Ou bien x est accepté par M_1 et, dans ce cas, la machine M' s'arrêtera en acceptant à l'itération i, si i est le nombre d'étapes de calcul de M_1 sur x. Ou bien x n'est pas accepté par M_1 et dans ce cas M' ne s'arrête pas.
  Ainsi, dans le cas où M ne s'arrête pas sur w, $L(M') = L_1$.
- Si M s'arrête sur w après n étapes de calcul, alors ou bien M_1 s'arrête sur x en moins de n étapes et M' s'arrête en **accept** (ligne 6). Ou bien M_1 ne s'arrête pas en moins de n étapes sur x et, dans ce cas, M' accepte x ssi $x \in L_2$.
  En fin de compte, dans ce cas :

$$\begin{aligned} L(M') = \quad & \{x \in L_1 \mid M_1 \text{ s'arrête en moins de } n \text{ étapes sur } x\} \\ & \cup \{x \in L_2 \mid M_1 \text{ ne s'arrête pas en moins de } n \text{ étapes sur } x\} \end{aligned}$$

  Mais comme $L_1 \subseteq L_2$ et que M_1 ou bien s'arrête en moins de n étapes sur x, ou bien le contraire, $L(M') = L_2$.

Donc $L(M') \in \{L_1, L_2\}$, et $L(M') = L_1$ ssi M ne s'arrête pas sur w. Et donc $L(M') \in \mathcal{P}$ ssi M ne s'arrête pas sur w.

2. Montrer que que s'il existe $L \in \mathcal{P}$ tel que, pour tout $L' \subseteq L$, ou bien L' est infini, ou bien $L' \notin \mathcal{P}$, alors $\mathcal{P}$ n'est pas récursivement énumérable.

Solution:
On réduit à nouveau le problème du non-arrêt de M sur w. Soit $L \in \mathcal{P}$ et supposons que L ne contienne pas de langage fini dans $\mathcal{P}$. On construit alors la machine M' comme suit :

Algorithm 2: M'

```
for i = 0 to +∞ do
    Simuler i étapes de calcul de M sur w
    if M ne s'arrête pas en moins de i étapes then
        Simuler i étapes de calcul de M_L sur x
        if M_L accepte x en moins de i étapes et |x| ≤ i then
            accept
        end
    end
end
```

Comme précédemment, si M ne s'arrête pas sur w, alors $L(M') = L$. Sinon, si M s'arrête en n étapes sur w, alors

$$L(M') = \{x \in L \mid M_L \text{ s'arrête sur } x \text{ en moins de } n \text{ étapes de calcul}\}$$

Dans ce dernier cas, $L(M') \subseteq L$ et $L(M')$ est fini puisque ses éléments sont tous de longueur $\leq n$. Par conséquent, par hypothèse, $L(M') \notin \mathcal{P}$ lorsque M s'arrête sur w.

3. On considère une application c (calculable) injective qui associe à chaque sous-ensemble fini de Σ^* un mot de Σ^*.[2] Montrer que, si $\mathcal{P}$ est récursivement énumérable, $\mathsf{Fin}(\mathcal{P}) = \{c(L(M)) \mid \langle M \rangle \in \mathcal{P}, L(M) \text{ fini}\}$ est récursivement énumérable.

Solution:
(sketch) Soit $M_{\mathcal{P}}$ la machine qui accepte $\mathcal{P}$. Étant donné $c(L)$, on calcule une machine M qui s'arrête toujours et accepte L, puis on simule $M_{\mathcal{P}}$ sur $\langle M \rangle$. Le langage ainsi accepté est l'ensemble des codes des langages finis de $\mathcal{P}$.

4. En déduire que $\mathcal{P}$ est récursivement énumérable si et seulement si les trois propriétés suivantes sont satisfaites :
   (a) $\mathsf{Fin}(\mathcal{P})$ est récursivement énumérable.
   (b) Tout langage de $\mathcal{P}$ contient un langage fini dans $\mathcal{P}$.
   (c) Pour tous langages $L \subseteq L'$ récursivement énumérables, si $L \in \mathcal{P}$ alors $L' \in \mathcal{P}$.

Solution:
D'après les 3 questions précédentes, il suffit de montrer que, quand les trois propriétés sont vérifiées, alors $\mathcal{P}$ est récursivement énumérable. On esquisse ici la construction de la machine de Turing sur la donnée $\langle M_L \rangle$:

2. Par exemple, $c(\{w_0, \ldots, w_n\}) = w_0 \# w_1 \ldots \# w_n$ où $w_0 <_{\mathsf{lex}} \cdots <_{\mathsf{lex}} w_n$

Algorithm 3: M'

```
1 for i = 1 to +∞ do
2     Considérer les i premiers mots des i premiers (codes des) langages finis de P
      (possible par la première propriété, avec une machine qui s'arrête toujours)
3     Pour chacun de ces mots, tester s'ils sont acceptés par M_L en moins de i
      étapes
```

Si L contient une langage fini de $\mathcal{P}$, il existera un entier i qui permettra de conclure que L contient un tel langage fini (i est le max du cardinal du langage fini, du numéro du langage fini, et du nombre maximal d'étapes de M sur chacun de ces mots).

Sinon, ou bien on épuise l'ensemble des langages finis de $\mathcal{P}$ et on rejette, ou bien on ne s'arrête pas.

Dans tous les cas, M' accepte $\langle M_L \rangle$ ssi L contient un langage fini de $\mathcal{P}$. Or les conditions (b) et (c) montrent que $\langle M_L \rangle \in \mathcal{P}$ ssi L contient un langage fini de $\mathcal{P}$.

5. Montrer que les propriétés suivantes des langages récursivement énumérables ne sont pas récursivement énumérables :

   (a) $\mathcal{P} = \{\emptyset\}$

   Solution:
   La propriété (c) n'est pas respectée pour $L = \emptyset$ et $L' = \Sigma^*$.

   (b) $\mathcal{P} = \{\Sigma^*\}$

   Solution:
   La propriété (b) n'est pas respectée.

   (c) $\mathcal{P}$ est l'ensemble des langages finis.

   Solution:
   La propriété (c) n'est pas respectée pour L un langage fini et $L' = \Sigma^*$.

   (d) $\mathcal{P}$ est l'ensemble des langages réguliers.

   Solution:
   La propriété (c) n'est pas respectée pour $L = \emptyset$ et $L' = \{a^n b^n \mid n \in \mathbb{N}\}$.

   (e) $\mathcal{P}$ est l'ensemble des langages récursifs.

   Solution:
   La propriété (c) n'est pas respectée pour $L = \emptyset$ et $L' = L_{arret}$.

   (f) $\mathcal{P}$ est l'ensemble des langages récursivement énumérables, mais pas récursifs.

   Solution:
   La propriété (c) n'est pas respectée pour $L = L_{arret}$ et $L' = \Sigma^*$.