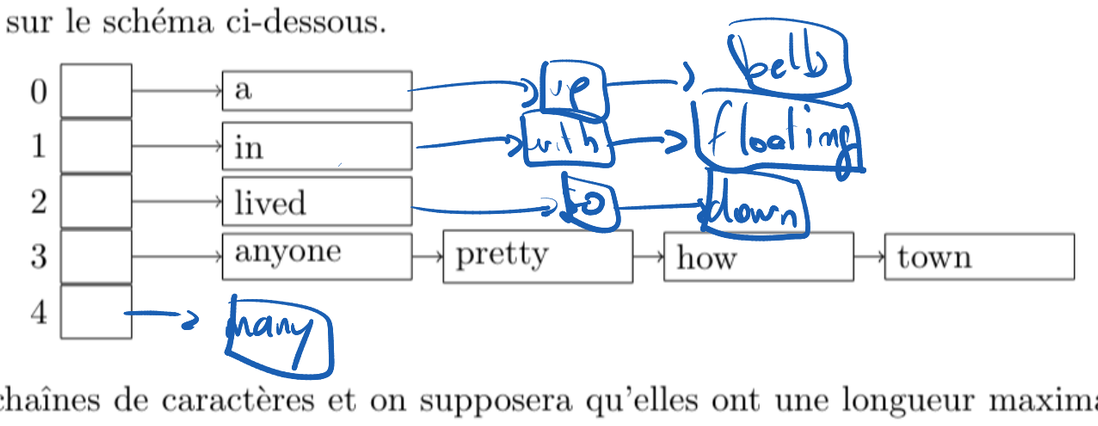
TDC: ASD.

Exercice 2 : Résolution par chaînage

On suppose disposer d'une table de hachage de taille $M = 5$ où ne sont stockées que des clés dont le contenu est décrit sur le schéma ci-dessous.



Les clés sont des chaînes de caractères et on supposera qu'elles ont une longueur maximale de 10. Les clés sont complétées à droite par des espaces si elles ne font pas la longueur maximale. La fonction transformant la clé en un entier est obtenue en réalisant la somme des rangs de chaque caractère de la clé dans la table ASCII[1].

Q 2.1 Compléter la table ci-dessus avec les mots : with, up, so, floating, many, bells, down.

Q 2.2 Discuter comment vont se répartir les clés lors de l'insertion dans la table lorsque :
- $M = 10$ et $h(k)$ est le reste de la division de la longueur de k par 10, -> tt le temps de longueur 10.
- $M = 128$ et $h(k)$ est le résultat de la fonction `ord` appliquée au dernier caractère de k, - mots de longueur < 10 ensemble
- $M = 10$ et $h(k)$ est le reste de la division par 10 de la somme des rangs des caractères élevée au carré. -> forcément des collisions.

with = 635 so = 482 many = 629

up = 485 floating = 916 bells = 690

down = 632

Exercice 3 : Le hachage qui fait coucou

Nous désirons stocker des éléments dans une table de hachage du coucou. Notre table contient dix cellules, nous allons la séparer en deux tables T_1 et T_2, de taille identique.

Nous souhaitons insérer les éléments suivants, dans l'ordre donné, dans notre table de hachage. Les résultats des fonctions de hachage h_1 et h_2 utilisées pour la table de hachage du coucou sont également indiquées.

k	17	59	31	75	66	76
h_1	1	3	1	3	1	3
h_2	4	2	2	1	1	0

Q 3.1 Quel est, après chaque étape, le contenu de T_1 et T_2 ?

T_1

	31		75						

T_2

76	66	59		17					

<u>Exercice 4</u> : Une utilisation des tables de hachages pour trier

Dans cet exercice, nous allons utiliser des tables de hachage à adressage fermé afin d'implanter un algorithme de tri. Contrairement au cours, on insèrera en queue de liste et non en tête de liste.

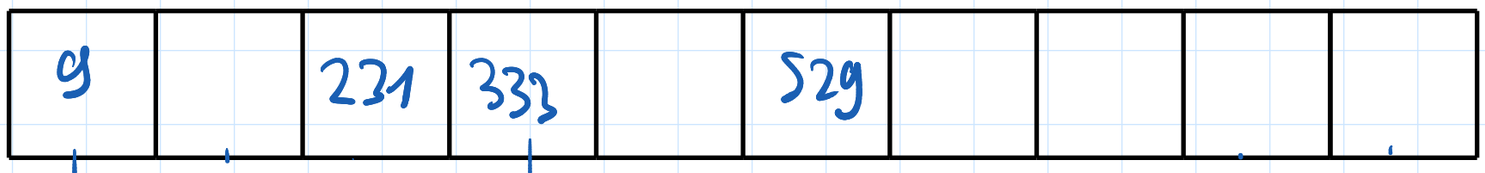
Nous cherchons à trier des matricules, des nombres décimaux mais écrits sur un nombre `MAX` **fixe** de chiffres (un type pour représenter un matricule pourrait être un tableau de `MAX` chiffres). On suppose disposer d'une fonction `nieme_chiffre` qui retourne le $n-$ème chiffre (en partant de la droite) d'un matricule. Par exemple, si `MAX` $= 8$, un matricule $m = 41143671$, `nieme_chiffre`$(m, 3) = 6$.

L'algorithme proposé est le suivant :

Entrée : d : un tableau contenant tous les matricules

pour *i allant de 1 à `MAX`* **faire**
  t : table de hachage de taille 10 dont la fonction de hachage est $h(k) =$ `nieme_chiffre`(k, i);
  pour chaque *matricule m de d* **faire**
    Ajouter m à t;
  fin
  Vider d;
  Ranger tous les matricules de t dans le tableau d en parcourant les alvéoles de 0 à 9 et les listes de chaque alvéole dans l'ordre;
fin
Imprimer le contenu de d;

Q 4.1 Tester l'algorithme sur les données suivantes : 231, 45, 529, 9, 67, 333, 51, 21. Pour chaque étape i, dessiner la table `t` et le tableau `d`.



21
↓
45
↓
51
↓
67

9, 21, 45, 51, 67, 231, 333, 529.