Mode T9

Projet de programmation fonctionnelle

Année 2024-2025

Préambule

- Pour chaque fonction, **le contrat**, les **tests unitaires** ainsi que son **code commenté** sont demandés.
- Vous êtes libres d'ajouter autant de fonctions auxiliaires que vous le jugez nécessaire. Elles devront également être commentées et testées.
- Le code rendu doit **impérativement compiler**.
- Interdiction de modifier les signatures (nom des fonctions + type) indiquées dans le sujet.
- La non utilisation d'itérateur sera pénalisée.
- L'utilisation non nécessaire d'accumulateurs (à l'aide de fonction auxiliaires) sera pénalisées.
- Vous êtes autorisé (et incité !) à utiliser toutes les fonctions de la librairie standard, en particulier le module List : `https://ocaml.org/api/List.html`.
- Pour tester dans utop, il faut faire **open** T9;; avant d'avoir accès aux modules Naif et Intuitif .
- Le projet est à faire en **binôme** et à rendre avant le **vendredi 27 juin à 18h**.
- Tous les projets seront testés pour toute forme de plagiat qui sera sanctioné par la note minimal et de possible sanctions disciplinaires.

Utilisation d'IA génératrice

Certains seront certainement tentés d'utiliser des IA génératrices pour réaliser le projet. C'est pour cela qu'une attention toute particulière sera portée à la qualité du code, la pertinence des commentaires ainsi qu'à la cohérence et à la couverture des jeux de tests.

Un code qui fonctionne n'est pas suffisant pour valider le projet. Il doit être lisible, compréhensible, commenté et bien testé.

1 Envoi de SMS avant l'air des smartphones

Les claviers des anciens modèles de téléphones portables (voir 1a) sont dotés de touches qui comportent plusieurs lettres (3 ou 4) (voir 1b). Pour envoyer des SMS, il faut appuyer (éventuellement plusieurs fois) sur les touches correspondants à chaque lettre du mot.

Sans saisie intuitive Pour saisir une lettre, il faut taper plusieurs fois sur une touche pour faire défiler les lettres correspondant à la touche. Par exemple, le c est situé sur la touche 2, la position du c est la troisième de abc, il faut donc appuyer 3 fois sur le 2 pour saisir un c. Donc pour écrire `bonjour` il faut appuyer : 2 fois sur 2 ; 3 fois sur 6 ; 2 fois sur 6 ; 1 fois sur 5 ; 3 fois sur 6 ; 2 fois sur 8 ; 3 fois sur 7.

Au total, il faut appuyer **16 fois** sur les touches pour écrire ce mot.

(a) Téléphone à touches

(b) Zoom sur les touches

FIGURE 1 – Les téléphones avant les smartphones

Avec saisie intuitive : mode T9 Le mode T9 (text on 9 keys) facilite la saisie des textes. Dans le mode T9, une seule frappe par lettre suffit et le téléphone propose de lui-même les mots, à partir d'un dictionnaire, qui correspondent à la séquence de touches qui vient d'être tapée.

Par exemple, la suite de touches 2, 6, 6, 5, 6, 8 et 7 correspond au mot `bonjour`. Cependant, il est possible qu'une suite de touches corresponde à plusieurs mots. Ainsi, la suite 8, 3, 6, 3, 7, 3 correspond aux mots `tendre` et `vendre`.

2 Description des fichiers fournis

- `chaines.mli` et `chaines.ml` : identiques aux fichiers fournis pour le TP sur les arbres lexicographiques, pour décomposer et recomposer des chaînes de caractères.
- `encodage.ml` : définition d'un type associant à chaque touche du clavier numérique un ensemble de lettres. Deux configurations du clavier numérique sont proposées : celle correspondant au mode T9 et une où les voyelles sont associées à la touche 2 et les consonnes à la touche 3.
- `naif.ml` : à compléter avec les fonctions correspondant à la saisie sans mode intuitif.
- `intuitive.ml` : à compléter avec les fonctions correspondant à la saisie intuitive.

3 Sans saisie intuitive

Comme indiqué précédemment, sans saisie intuitive, il faut appuyer plusieurs fois sur une touche pour saisir une lettre. Quand deux lettres successives correspondent à la même touche, c'est une pause dans la saisie qui indique le passage à la lettre suivante. Pour faciliter l'écriture des fonctions, nous remplaçons cette pause par un appui sur la touche 0. Ainsi pour saisir "bonjour", il faudra la suite de touches 2, 2, 0, 6, 6, 6, 0, 6, 6, 0, 5, 0, 6, 6, 6, 0, 8, 8, 0, 7, 7, 7, 0.

▷ **Exercice 1** ***Mots → touches***

1. *Écrire la fonction* encoder_lettre : encodage -> char -> (int * int) *qui indique la touche et le nombre de fois qu'il faut appuyer dessus pour saisir la lettre passée en paramètre. La liste associative (touche, liste de lettres) est passée en paramètre de la fonction.*
2. *Écrire la fonction* encoder_mot : encodage -> string -> int list *qui calcule la suite de touche à presser pour saisir un mot passé en paramètre. La liste associative (touche, liste de lettres) est passée en paramètre de la fonction.*

▷ **Exercice 2** ***Touches → mot***

1. *Écrire la fonction* decoder_lettre : encodage -> int * int -> char *qui identifie la lettre saisie à partir d'une touche et du nombre de fois qu'elle a été pressée. La liste associative (touche, liste de lettres) est passée en paramètre de la fonction.*
2. *Écrire la fonction* decoder_mot : encodage -> int list -> string *qui identifie le mot saisi à partir d'une suite de touches. La liste associative (touche, liste de lettres) est passée en paramètre de la fonction.*

4 Avec saisie intuitive

▷ **Exercice 3** ***Mots → touches***

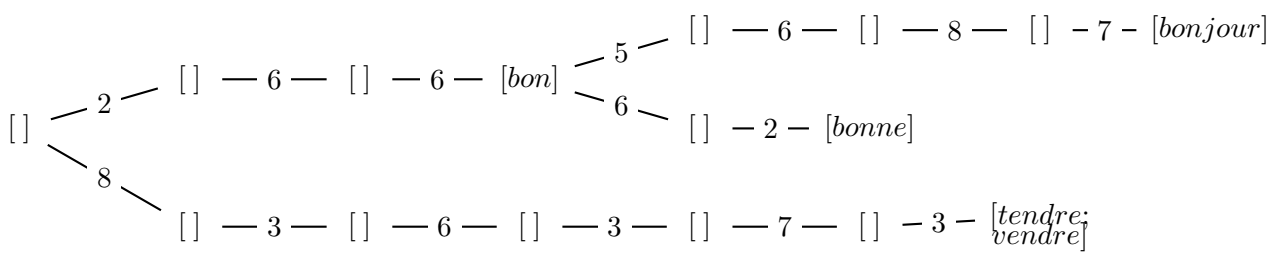
1. *Écrire la fonction* encoder_lettre : encodage -> char -> int *qui indique la touche associée à la lettre passée en paramètre. La liste associative (touche, liste de lettres) est passée en paramètre de la fonction.*
2. *Écrire la fonction* encoder_mot : encodage -> string -> int list *qui calcule la suite de touches à presser pour saisir un mot passé en paramètre. La liste associative (touche, liste de lettres) est passée en paramètre de la fonction.*

Dictionnaire Comme indiqué précédemment, le mode de saisie intuitive repose sur un dictionnaire qui à une suite de touche associe un ensemble de mots.

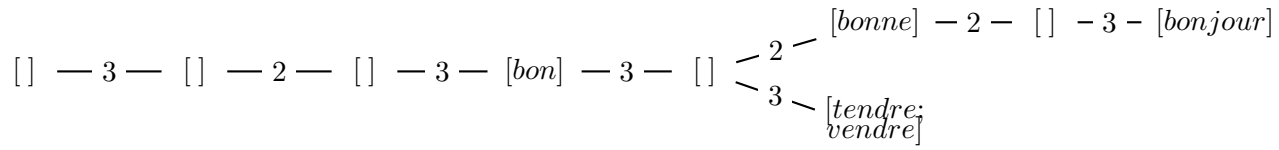
Ce dictionnaire est représenté par un arbre n-aire avec des listes de mots dans les nœuds et des chiffres (touches) sur les branches :

type dico = Noeud **of** (string list * (int * dico) list)

Pour le mode T9, le dictionnaire composé des mots "bon", "bonne", "bonjour", "vendre" et "tendre" est le suivant :



Pour le mode voyelles / consonnes, le dictionnaire composé des mêmes mots est le suivant :



Même si c'est le cas dans les exemples, il n'est pas demandé que les branches soient ordonnées.

Pour les tests, une liste de mots du langage français est fournie dans `dico_fr.txt` (Petit Larousse Illustré 2007).

▷ **Exercice 4** ***Manipulation d'un dictionnaire***

1. *Écrire la constante* empty : dico *qui créer un dictionnaire vide.*
2. *Écrire la fonction* ajouter : encodage –> dico –> string –> dico *qui ajoute un mot à un dictionnaire. La liste associative (touche, liste de lettres) est passée en paramètre de la fonction.*
3. *Écrire la fonction* creer_dico : encodage –> string –> dico *permettant de construire un dictionnaire à partir d'un encodage et d'un fichier (le second paramètre de la fonction est son chemin d'accès). Aidez-vous de la documentation en ligne pour la lecture de fichier. Exceptionnellement, des traits d'impératifs sont autorisés pour cette fonction. Elle n'est pas bloquante pour les fonctions suivantes mais peu faciliter l'écriture des tests.*
4. *Écrire la fonction* supprimer : encodage –> dico –> string –> dico *qui supprime un mot à un dictionnaire. La liste associative (touche, liste de lettres) est passée en paramètre de la fonction. Cette fonction devra élaguer les branches éventuellement devenues inutiles du dictionnaire.*
5. *Écrire la fonction* appartient : encodage –> dico –> string –> bool *qui vérifie si un mot appartient au dictionnaire.*
6. *Un dictionnaire n'est cohérent pour un encodage donné, que si tous les mots dans un nœud sont cohérents avec le chemin qui y mène. Écrire la fonction* coherent : encodage –> dico –> bool *qui vérifie si un un dictionnaire est cohérent pour un encodage donné.*

▷ **Exercice 5** ***Récupération d'informations sur un dictionnaire***

1. *Écrire la fonction* decoder_mot : dico –> int list –> string *qui identifie l'ensemble des mots correspondant à une suite de touches dans un dictionnaire.*
2. *Il peut être intéressant de proposer une liste de mot à l'utilisateur avant qu'il ait fini de saisir son mot. Écrire la fonction* prefixe : dico –> int list –> string list *qui liste l'ensemble des mots d'un dictionnaire dont le préfixe a été saisi.*
3. *Écrire la fonction* max_mots_code_identique : dico –> int *qui calcule le nombre maximal de mots ayant la même séquence de touches dans un dictionnaire.*

4. *Écrire la fonction* `lister : dico -> string list` *qui liste l'ensemble des mots d'un dictionnaire.*
5. *Dans le cas où la saisie n'est associée à aucun mot, il est probable que l'utilisateur ait fait une faute de frappe. Écrire la fonction* `proche : dico -> int list -> int -> string list` *qui identifie l'ensemble des mots correspondant à une suite de touches dans laquelle exactement* `n` *(troisième paramètre) erreurs se sont glissées.*