(English will follow)

Mettez en place votre propre dictionnaire (avec beaucoup plus de fonctionnalités que le dictionnaire traditionnel!)

La principale fonction est évidemment de rechercher la signification d'un mot particulier et son type nom/verbe, etc. Dans de nombreux dictionnaires, lorsque nous entrons un mot erroné, le message "aucun résultat trouvé" s'affiche, **mais vous allez implémenter une fonction qui affiche les mots similaires au mot entré disponibles dans le dictionnaire *s'ils ne sont pas trouvés***. Cette fonction peut être mise en œuvre de différentes manières, mais vous utiliserez la correspondance de préfixes.

Mise en œuvre :

- Un historique de recherche. Il aide l'utilisateur à mémoriser les mots car il arrive parfois que l'on oublie les mots que l'on a recherchés auparavant, mais que l'on veuille en rechercher à nouveau la signification parce qu'on l'a oublié
- Un traducteur de l'anglais au français.

Notez que,

- vous devrez utiliser le CSV fourni avec 5 588 mots anglais, leur type (nom, préposition, verbe, etc.), leur signification et leur traduction française ;
- vous devrez créer un menu simple dans la console pour chacune des fonctionnalités mentionnées ci-dessus, c'est-à-dire pour 1) rechercher le mot 2) imprimer l'historique 3) traduire le mot (comme vous l'avez fait dans le TP1)

Conseils :

- Recherche de préfixe :
    - Si un mot n'est pas présent dans le dictionnaire, recherchez la sous-chaîne de longueur maximale depuis le début qui est présente dans le dictionnaire, et renvoyez tous les mots commençant par ce préfixe dans le dictionnaire.
    - Exemple : Le dictionnaire contient 7 mots - "apple, apply, above, zebra, apartment, able, app". La recherche du mot "aplpe" devrait renvoyer "apple, apply, apartment, app", car ces 4 mots ont la plus longue sous-chaîne "ap" présente dans le mot de la requête.
- Explorez la structure de données TRIE pour mettre en œuvre toutes les fonctionnalités susmentionnées (section 13.3 du manuel de référence).

Fichier de test :

- Les fichiers testPrefix.txt et testPrefixResult.txt vous ont été fournis pour vous permettre de vérifier l'implémentation par vous-même. Écrivez un code capable de lire un fichier et de générer le fichier texte de sortie comme indiqué ci-dessous.
- testPrefix.txt contient un mot dans chaque nouvelle ligne à rechercher dans le dictionnaire.
- testPrefixResult.txt contient le nombre total de mots similaires pour le mot recherché.

Exemple : Pour le fichier de test donné ci-dessous

testPrefix.txt

App
Againx
Fom

Le fichier de sortie généré par votre code devrait être le suivant :

testPrefixResult.txt

2697
7
4

Implement your own Dictionary (*with many more features than the traditional one!*)

The main feature is obviously to search for a meaning of a particular word and their type noun/verb etc. In many of the dictionaries when we enter a wrong word it shows “no results found”, but you will implement a function that **displays words similar to the entered word available in the dictionary *if they are not found***. This can be implemented in various ways but you shall use Prefix matching.

Implementation:

- A search history. It helps the user to memorize the words as sometimes it happens that you forgot the words you searched for before, but you want to search the meaning again because you forgot it;
- A translator from English to French.

Note that,

- you will need to use the provided CSV with 5,588 English words, their type (noun, preposition, verb, etc.), meaning, and French translation;

- you should create a simple menu in consol for each of the above-mentioned features i.e. for 1) search word 2) print history 3) translate the word (the way you made it in TP1)

Hints:

- Prefix search:
    - If a word is not present in the dictionary then look for the maximum length substring from the start which is present in the dictionary, and return all the words starting with that prefix from the dictionary
      Example:
    - The dictionary has 7 words - "apple, apply, above, zebra, apartment, able, app". The search for the word "aplpe" should return "apple, apply, apartment, app"; because these 4 words have the longest substring "ap" present from the query word
- Explore TRIE data structure to implement all of the above-mentioned features (coursebook reference section 13.3)

Test file:

- You have been provided with the testPrefix.txt and testPrefixResult.txt file for checking the implementation on your own. Write a code which can read a file and generate the output text file as shown below.
- **testPrefix.txt** contains a word in each new line to search in the dictionary
- **testPrefixResult.txt** contains a total number of similar words for the respective searched word

Example: For the test file given below

testPrefix.txt

App
Againx
Fom

The output file generated by your code should be as follows:

testPrefixResult.txt

2697
7
4