**IFT-2008 – Algorithmes et structures de données**
**HIVER 2022**

# Travail pratique 3 (individuel)

À remettre, par le portail du cours avant 17h00 le vendredi 22 avril 2022

## 1 Mise en contexte

Nous vous demandons de réaliser un petit programme qui traduit un texte de l'anglais au français ainsi qu'une correction orthographique si le mot n'est pas trouvé dans le dictionnaire. La langue de base du dictionnaire est donc l'anglais. Le programme doit d'abord lire le fichier qui contient le dictionnaire anglais-français, puis stocker l'information de ce fichier dans un arbre AVL. Après avoir lu le dictionnaire, le programme demande un texte en anglais à l'utilisateur. Ensuite, pour chaque mot du texte, le programme recherche ce mot dans le dictionnaire et traduit ce mot en français. Si le mot n'est pas trouvé dans le dictionnaire, une liste d'un maximum de 5 mots semblables au mot entré est proposée à l'utilisateur en calculant une certaine similitude entre les mots. L'utilisateur choisit l'un de ces mots, puis le programme traduit le mot. Si aucune traduction ne se trouve dans l'arbre, l'utilisateur est averti et le mot reste inchangé. Par exemple :

Entrez le nom du fichier du dictionnaire Anglais-Français : EnglishFrench.txt
Entrez un texte en anglais (pas de majuscules ou de ponctuation/caracteres speciaux) :
this homeworrk is funny
Le mot 'homeworrk' n'existe pas dans le dictionnaire. Veuillez choisir une des suggestions suivantes :
1. homework
2. homeworker
3. homeworking
4. homestead
5. homey
Votre choix : 1
Plusieurs traductions sont possibles pour le mot 'homework'. Veuillez en choisir une parmi les suivantes :
1. devoir
2. travail
3. exercice
Votre choix : 2
Plusieurs traductions sont possibles pour le mot 'funny'. Veuillez en choisir une parmi les suivantes
1. drole
2. amusant
3. plaisant
4. agreable
Votre choix : 2
La phrase en francais est :
ce travail est amusant

## 2 Modèle d'implantation

La classe Dictionnaire qui modélise notre dictionnaire de traduction se trouve dans le fichier Dictionnaire.h. Notez que la base du dictionnaire est la langue anglaise. Ainsi, chaque nœud de l'arbre AVL contient un mot en anglais. La plupart des méthodes du dictionnaire requièrent de la récursivité et feront donc des appels initiaux à une autre méthode, cette fois privée, que vous développerez (et dont vous déciderez du prototype) qui, elle sera récursive. Vous pouvez écrire d'autres méthodes en plus de celles demandées (publiques ou privées). Voici un extrait du dictionnaire anglais-français fourni :

…
Jewelry bijouterie[Noun]
jewels joyaux[Noun]
jib (of crane) flèche (f)[Noun]
jib (sail) foc (m)[Noun]
jiffy (in a ~) en un clin d'oeil
jig gigue (f)[Noun]
jigsaw (puzzle) puzzle (m)[Noun]
jihad lutte, combat (Islam)[Noun]
jilt laisser tomber[Verb]
jingle (bell) tinter[Verb]
jingle (song) jingle (m), indicatif (m)[Noun]
jingle (sound) cliquetis (m)[Noun]
jinx poisse (f)[Noun]
jitters trac[Noun]
job boulot (colloq.)[Noun]
job emploi (m)[Noun]
job turbin (slang)[Noun]
jobless au chômage[Adjective]
…

Nous vous fournissons le constructeur du dictionnaire qui va lire à partir du fichier texte fourni les mots ainsi que leurs traductions en ignorant les fonctions des mots spécifiées entre crochets ([Noun], [Verb], etc.) et les contextes spécifiés entre parenthèses ((bell), (song), etc.). En fait, on ne considère que la première des traductions sur chaque ligne. Par exemple, pour le mot jingle, le constructeur appellera la méthode ajouteMot() 3 fois. Une fois avec la traduction tinter, une fois avec jingle, et une fois avec cliquetis.

Nous vous fournissons également la surcharge de l'opérateur d'affichage (implanté *inline*) permettant d'afficher le dictionnaire niveau par niveau. Le correcteur va utiliser entre autres cette méthode pour vérifier que l'arbre binaire de recherche est bien balancé.

Par ailleurs, nous utilisons la classe interne *NoeudDictionnaire* représentant le nœud typique de l'arbre binaire de recherche. Elle contient le mot en anglais, ses traductions (un vector de chaînes de caractères) et bien sûr le pointeur sur l'arbre gauche, le pointeur sur l'arbre droit et la hauteur du nœud (utilisé pour équilibrer l'arbre). Il est évidemment interdit de modifier ce modèle d'implantation.

## 3 Travail à faire

Il s'agit d'implémenter dans le fichier dictionnaire.cpp toutes les méthodes décrites dans le fichier d'interface dictionnaire.h. Vous allez aussi devoir ajouter ou enrichir les commentaires d'interface et d'implémentation

en format *Doxygen* surtout dans les fichiers.cpp. Nous vous suggérons très fortement de créer des fichiers permettant de tester séparément les méthodes que vous devez implanter. Vous devez donc faire tous les tests nécessaires à chaque étape importante dans votre implémentation afin de vous assurer que cette implantation est correcte au fur et à mesure de sa conception surtout concernant le balancement de l'arbre binaire de recherche lors de l'ajout et la suppression des mots dans le dictionnaire.

Il est à noter que la méthode similitude() prend deux mots en paramètre et les compare. Elle retourne un nombre entre 0 et 1. "1" signifie que les mots sont identiques, et "0" signifie que les mots sont complètement différents. Un nombre rapproché de 1 (par exemple 0.95) voudra dire que les mots sont très semblables (par exemple "analyser" et "analiser"), et un nombre rapproché de 0 (par exemple 0.04) voudra dire que les mots sont très différents (par exemple "ici" et "vêtements"). Libre à vous de décider comment implémenter cette mesure de similitude qui permet d'améliorer le correcteur d'orthographe si elle est efficace.

Pour vous aider, nous vous fournissons dans le fichier Principal.cpp une fonction main qui est incomplète. Elle permet d'ouvrir le fichier contenant le dictionnaire, de charger les mots en utilisant le constructeur fourni. C'est à vous donc de compléter ce fichier afin d'interagir avec l'utilisateur en lui offrant les différentes suggestions de traduction.

# 4 Ce que vous devez rendre

À l'aide de monportail, vous devez rendre un fichier .zip comportant **uniquement** les fichiers suivants (il est important d'ajouter les balises Doxygen dans le fichier Dictionnaire.cpp, votre nom dans la section « *author* » ainsi que des commentaires à l'intérieur des méthodes pour expliquer votre méthode d'implémentation et ainsi faciliter le travail de correction) :

- **Dictionnaire.h** contenant l'interface du type abstrait Dictionnaire.
- **Dictionnaire.cpp** représentant la source de votre implémentation.
- **Principal.cpp** contenant un menu, qui permet à un utilisateur d'entrer un texte en anglais et de le traduire en français.
- **EnglishFrench.txt** contenant le dictionnaire Anglais-Français.
- **NoteTp3-IFT2008.xls** (un fichier Excel contenant le barème de correction. Il faut juste ajouter votre nom et matricule sur la première ligne).

**Vous ne devez en aucun cas ajouter un autre fichier ou répertoire. N'incluez aucun exécutable, ni aucun fichier généré par votre environnement de développement (pas de répertoire debug s.v.p).** De plus, vous n'avez pas à générer la documentation Doxygen en format html pour l'ensemble du projet (pour ne pas envoyer un gros fichier zip). Le correcteur pourra le faire lors de la correction.

**Attention !** Il est **formellement interdit de partager ou publier** le code du TP ou votre solution sur le Web avant ou après la remise du TP, car c'est une source évidente de plagiat et vous risquez donc d'être pénalisé. De plus, **tout travail remis en retard se verra pénalisé de -25% de la note**. Le retard débute dès la limite de remise dépassée (dès la première minute). Un retard excédant une journée provoquera le rejet du travail pour la correction et la note de 0 pour ce travail. Le nom du .zip doit respecter le format suivant : **TP3-Matricule.zip**. Je vous rappelle également qu'il est de votre responsabilité de vérifier après la remise que vous nous avez envoyé les bons fichiers (**non vides et non corrompus**), sinon vous pouvez avoir un zéro.

**PS.** Vous devez considérer les mêmes remarques importantes du TP1 concernant les normes de programmation, la tolérance zéro vis-à-vis des erreurs de compilation et la portabilité des programmes.

**Bon travail.**