L3 APP **Lahloukarim@free.fr**	**Septembre 2018** **Programmation Langage C/C++** **Abdelkrim LAHLOU**

TP- 2 : Les Tableaux, les chaines de caractères et fonctions

Loi de Zipf :

La loi de Zipf est une loi empirique que les textes respectent plus au moins. On cherche d'abord à classer les mots par fréquence : on compte le nombre de fois où chaque mot apparaît dans le texte, ce nombre est nommé sa **fréquence**. Ensuite, on classe les fréquences de tous les mots par ordre décroissant, et on devrait observer que la deuxième fréquence est la moitié de la première, que la troisième est la moitié de la deuxième, et ainsi de suite.

Recherche d'un mot quelconque dans un texte.

Quelques précautions avant de traiter le texte :

On considère que le texte que l'on veut analyser pour vérifier la pertinence de la loi de Zipf est obtenue par un simple copier/coller à partir d'une page web ou d'un document électronique quelconque. Une partie du travail d'analyse va consister à découper le texte en mots pour pouvoir réaliser leur comptage. Comment peut-on repérer qu'un ensemble de caractères constitue un mot ?

Exemple pour résoudre cette question : dans le texte suivant, quels sont les mots présents et quels sont leur(s) fréquences respectives ?

```
"Une personne m'a dit le mot bonjour. J'ai répondu bonjour à cette personne! Bonjour? Quel joli mot!"
```

mots et sous-mots.

Lors de la recherche d'un mot, il faut bien faire attention à ne pas compter les occurrences d'un mot lorsqu'il est inclus dans un autre mot. Comment être sûr de ne compter que les mots isolés et, pour l'exemple suivant, de ne compter le mot 'café' que 2 fois.

```
"sais-tu où je peux prendre un café ? oui, pour un café, il faut aller à la cafétéria !"
```

a) écrire un programme qui recherche, dans un tableau de caractères, le nombre de fois où apparaît un mot quelconque (qui peut être saisi au clavier).

b) constitution d'un dictionnaire : on veut garder une liste des mots rencontrés lors du parcours du texte, sans forcément garder leur ordre, mais en ayant un seul exemplaire de chaque mot du texte : le résultat global que l'on cherche à obtenir, sur le texte exemple suivant, est :

```
"Une personne m'a dit le mot bonjour. J'ai répondu bonjour à cette personne! Bonjour? Quel joli mot!"
```

dictionnaire : "Une – personne – m'a – dit – le – mot – bonjour – j'ai – répondu – à – cette – quel – joli"

fréquences (dans le même ordre que les mots du dictionnaire) : 1 2 1 1 1 2 3 1 1 1 1 1 1

Améliorer le programme pour qu'il "supprime" du tableau contenant le texte à traiter le mot recherché et en range un exemplaire dans un nouveau tableau où seront stockés les mots en un exemplaire (ce nouveau tableau sera le dictionnaire).

c) Pour supprimer un mot du tableau contenant le texte d'origine, on remplace toutes ses lettres par le caractère '#'.

Illustration : dans le texte "le magasin ouvrira le 12 juin et le 14 juin.", on compte 3 fois le mot "le". Le but est d'arriver au texte :

"## magasin ouvrira ## 12 juin et ## 14 juin.", et de stocker le mot "le" dans le dictionnaire.

De même, le texte contient 2 fois le mot "juin". Le but est d'arriver au texte :

"## magasin ouvrira ## 12 #### et ## 14 ####.", et au dictionnaire

"le juin"

d) A partir des questions a et b, on peut calculer la fréquence de chaque mot du texte. Ecrire un programme qui permet de calculer ces fréquences et qui les range dans un tableau d'entiers. Il faudra parcourir le tableau contenant le texte d'origine pour y détecter les différents mots et les compter.

Pour vous aider à déterminer l'algorithme, répondez à la question suivante : lorsque l'on commence à traiter le texte, on détecte le premier mot du texte, puis on le recherche dans tout le reste du texte. Lorsque cette recherche est terminée, on passe au traitement du deuxième mot du texte. Est-il alors nécessaire de recommencer la recherche depuis le début du tableau ?

e) Améliorer le programme pour qu'il affiche : les fréquences dans l'ordre décroissant et les mots associés à chacune des fréquences.