Option Informatique en Spé MP et MP*
Découpage d'un paragraphe en lignes

▶ Nous nous proposons de décrire plusieurs algorithmes de découpage d'un paragraphe en lignes et de justification de celles-ci, en vue de leur impression ou de leur affichage sur un écran. Un paragraphe sera fourni sous forme d'une chaîne de caractères (type `string`) ; on rappelle qu'en Caml, l'indexation des caractères d'une chaîne commence à 0 : si `s` est de type `string`, la longueur de `s` est donnée par `string_length s` ; le premier caractère est donné par `s.[0]` et le dernier par `s.[string_length s - 1]`.

▶ L'emploi de références est interdit. Le temps d'exécution de chaque fonction devra être un grand $\mathcal{O}$ de la taille de ses arguments ; par exemple, le découpage en lignes d'un paragraphe devra s'effectuer en un temps linéaire par rapport à sa longueur. Lorsque l'on vous demande d'*imprimer*, vous devrez utiliser les fonctions `print_string: string -> unit` et/ou `print_newline: unit -> unit`.

▶ Un paragraphe est formé de *mots* séparés par des *espaces* : un mot est une suite de caractères ne contenant aucun espace, qui commence au début du texte ou est précédée d'au moins un espace, et se termine à la fin du texte ou est suivie d'au moins un espace. Par exemple, le paragraphe suivant (encadré pour faciliter la compréhension) :

```
Ceci n'est qu'un essai.
```

comporte cinq mots, de longueurs respectives 4, 5, 5 et 6 (le point compte dans le dernier mot). Nous supposons pour l'instant que tous les caractères ont la même largeur, égale à celle d'un espace.

Question 1 • Rédigez en Caml une fonction de signature

```
decompose_en_mots: string -> string list
```

spécifiée comme suit : `decompose_en_mots p` décompose le paragraphe p en mots. Par exemple :

```
decompose_en_mots "Ceci n'est qu'un essai.";;
```

rendra la liste `["Ceci";"n'est";"qu'un";"essai."]`.

▶ Le premier algorithme à mettre en œuvre est l'algorithme *glouton* : nous plaçons sur chaque ligne le maximum de mots consécutifs ; lorsqu'un mot ne peut tenir sur la ligne courante, nous passons à la ligne suivante.

Question 2 • Rédigez en Caml une fonction de signature

```
prend_une_ligne: int -> string list -> string list*string_list
```

spécifiée comme suit : `prend_une_ligne n l` décompose la liste de mots $\ell = (m_1, m_2, \dots, m_q)$ en un couple (ℓ_1, ℓ_2) défini comme suit : soit p le plus grand indice vérifiant $\sum\limits_{1 \leqslant i \leqslant p} |m_i| \leqslant n - p + 1$; alors $\ell_1 = (m_1, \dots, m_p)$ et $\ell_2 = (m_{p+1}, \dots, m_q)$ (étant entendu que ℓ_2 est la liste vide si $p = q$). Par exemple :

```
prend_une_ligne 12 ["Ceci";"n'est";"qu'un";"petit";"essai."];;
```

rendra le couple `(["Ceci";"n'est"],["qu'un";"petit";"essai."])`.

Question 3 • Rédigez en Caml une fonction de signature

```
decoupe_en_lignes: int -> string -> string list list
```

spécifiée comme suit : `decoupe_en_lignes n p` découpe le paragraphe p en lignes de largeur au plus égale à n. Par exemple :

```
decoupe_en_lignes 13 "Ceci n'est qu'un tout petit essai.";;
```

rendra la liste `[["Ceci";"n'est"];["qu'un";"tout"];["petit";"essai."]]`.

Question 4 • Donnez une condition nécessaire et suffisante portant sur p et n pour que ce découpage soit possible.

Question 5 • Rédigez en Caml une fonction de signature

```
imprime_ligne: string list -> unit
```

spécifiée comme suit : `imprime_ligne l` imprime les mots qui constituent la liste ℓ, en insérant un espace exactement entre deux mots consécutifs.

Question 6 • Rédigez en Caml une fonction de signature

```
imprime_texte: int -> string -> unit
```

spécifiée comme suit : `imprime_texte n p` imprime le paragraphe p, après l'avoir découpé en lignes pouvant tenir dans une largeur n au moyen de l'algorithme glouton.

► Le résultat fourni par `decoupe_en_lignes` n'est pas satisfaisant, esthétiquement : le bord droit du texte est irrégulier. Nous allons donc lui appliquer une *justification*, opération qui consiste à ajouter au besoin des espaces supplémentaires entre les mots de chaque ligne pour amener sa largeur totale à la valeur n exactement. L'objectif est l'écriture d'une fonction :

```
justifie_ligne: int -> string list -> string list
```

spécifiée comme suit : `justifie_ligne n l` justifie la ligne ℓ, en répartissant équitablement des espaces supplémentaires entre les mots qui la composent, pour amener sa longueur à n. Par exemple :

```
justifie_ligne 31 ["Ceci";"est";"un";"autre";"essai."];;
```

rendra la liste de chaînes suivante, dans laquelle nous avons matérialisé les espaces entre mots des étoiles :

```
["Ceci";"***";"est";"***";"un";"***";"autre";"**";"essai."]
```

Nous utiliserons le mot *insert* pour parler d'une suite d'espaces consécutifs séparant deux mots ; ainsi l'exemple précédent utilise quatre inserts, dont les trois premiers ont une largeur égale à 3 et le dernier a une largeur égale à 2.

► Soit $\ell = (m_1, m_2, \dots, m_q)$ une ligne (présentée comme liste de mots) que l'on souhaite justifier. Pour des raisons esthétiques, nous voulons que les espaces insérés entre les mots de cette ligne soient répartis le plus régulièrement possible ; plus précisément, les largeurs des inserts doivent, soit être toutes égales à une même valeur k, soit ne prendre que deux valeurs distinctes k et $k+1$

Question 7 • Exprimez k en fonction de n, du nombre q de mots de la ligne, et de la somme $v = \sum_{1 \leqslant i \leqslant q} |m_i|$ des longueurs de ces mots.

Question 8 • Rédigez en Caml une fonction de signature

```
repartit_espaces: int -> string list -> string list
```

spécifiée comme suit : `repartit_espaces n l` construit une liste $(e_1, e_2, \dots, e_{q-1})$ d'inserts permettant de justifier la ligne $\ell = (m_1, m_2, \dots, m_q)$ dans une largeur n.

Question 9 • Rédigez en Caml une fonction de signature

```
justifie_ligne: int -> string list -> string list
```

spécifiée comme suit : `justifie_ligne n l` construit la liste $(m_1, e_1, m_2, e_2, \dots, e_{q-1}, m_q)$ représentant la ligne justifiée.

Question 10 • Rédigez en Caml une fonction de signature

```
imprime_texte_justifie: int -> string -> unit
```

spécifiée comme suit : `imprime_texte_justifie n p` découpe le paragraphe p en lignes de longueur au plus n, les justifie et les affiche.

Question 11 • Soit $(m_i)_{1 \leqslant i \leqslant Q}$ la liste des mots d'un paragraphe p ; nous notons x_i la longueur du mot m_i. Donnez une condition *suffisante* portant sur n et sur la suite $(x_i)_{1 \leqslant i \leqslant Q}$ pour que l'impression justifiée soit réalisable au moyen de l'algorithme que l'on vient de mettre en œuvre.

Question 12 • Quel problème pose, en général, la dernière ligne d'un paragraphe ainsi composé ? Comment proposez-vous de le régler ?

FIN