TABLE DE CORRESPONDANCES

Objectif : Écrire un programme qui créer une table de correspondances des occurrences des mots contenus dans un fichier texte.

Contraintes :

1) Chaque entrée de la table correspond à un mot du texte et fait le lien avec la liste des numéros des lignes du fichier texte où ce même mot est apparu. Les mots se différencient par leur orthographe (Maison = maison). À une même orthographe correspond une seule entrée dans la table.

2) Après lecture du fichier, votre programme devra afficher le contenu de la table de correspondances.

3) Pour l'entrée des données on utilisera la redirection de l'entrée standard.

Exemple d'utilisation :

C :> Corresp < toto.txt

ou

$ Corresp < toto.txt

Le contenu du fichier « toto.txt » est alors placé dans l'entrée standard du programme « Corresp ».

Indications :

Pour stocker les mots et les numéros de ligne où ils apparaissent dans le fichier, on utilisera une structure `Entree` :

```
typedef struct
{
      char *mot;
      int lignes[50];
      int nbLignes;
}Entree;
```

La table de correspondance sera également une structure contenant un tableau d'entrées et le nombre d'entrées utilisées dans ce tableau :

```
#define MAX_ENTREES 500

typedef struct
{
      Entree entrees[MAX_ENTREES];
      int nbEntrees;
}Table;
```

Etant donné qu'une `Table` est une structure de taille importante (500 structures `Entree`), on préfèrera passer un pointeur sur `Table` en paramètre des fonctions plutôt qu'une `Table`. Voici un exemple avec la fonction qui initialise une `Table` :

```
void initTable(Table *t)
{
	t->nbEntrees = 0;
}
```

De plus, quand la Table est susceptible d'être modifiée par la fonction, le passage par pointeur est nécessaire.

Ecrire les fonctions suivantes, qui utilisent les déclarations ci-dessus :

- `void initEntree(Entree *t, char *mot, int numLigne);`
  Cette fonction est appelée quand un nouveau mot doit être inséré dans la table.
- `void ajouterLigne(Entree *t, int numLigne);`
  Cette fonction est appelée pour chaque nouvelle occurrence d'un mot qui appartient déjà à la table.
- `int rechercheEntree(Table *t, char *mot);`
  Cette fonction recherche un mot dans la table. Si le mot existe, la fonction renvoie l'indice de l'entrée correspondant à ce mot dans la table. Si le mot n'existe pas elle renvoie -1.
- `void traiterMot(Table *t, char * mot, int numLigne);`
  Cette fonction est appelée pour chaque mot du fichier texte. Elle ajoute une nouvelle entrée à la table si le mot n'y figure pas déjà. Sinon, elle met à jour l'entrée correspondante.
- `void afficheTable(Table *t);`
  Cette fonction affiche toutes les entrées de la table (mot + numéros de lignes).

Voici le programme principal qui, en utilisant les fonctions précédentes, construit la table de correspondance à partir du fichier texte et l'affiche :

```
int main()
{
	Table maTable;
	char c;
	char mot[80];
	int imot=0;
	int numLigne=0;
	initTable(&maTable);
	while((c=getchar())!=EOF)
	{
		if(isalpha(c))
			mot[imot++]=c;
		else
		{
			mot[imot]='\0';
			if(imot>0)
				traiterMot(&maTable,mot,numLigne);
			imot=0;
		}
		if(c=='\n')
			numLigne++;
	}
	afficheTable(&maTable);
	return 0;
}
```