Séance 3 - Tableaux

Le but de ce TP est de réaliser un jeu du pendu. On rappelle les règles du pendu : un mot est à deviner, et un joueur dispose d'un certain nombre de vies. Tant qu'il lui reste au moins une vie, le joueur peut proposer une lettre. Si la lettre figure dans le mot, alors il ne perd pas de vie, et l'emplacement de toutes les occurences de la lettre dans le mot lui sont données. Sinon, il perd une vie. Le joueur perd s'il n'a plus de vie, et gagne si toutes les lettres ont été découvertes.

Afin de réaliser ce programme, différentes fonctions vont devoir être codées au préalable.
Vous devez créer un seul projet Code : :Blocks pour tout le TP. Dans ce projet, comme pour le précédent TP, trois fichiers devront être créés :
– un fichier `main.c` qui contiendra la fonction main() seulement.
– un fichier `fonctions.h` qui contiendra tous les en-têtes (déclarations) de vos fonctions.
– un fichier `fonctions.c` qui contiendra tous les corps (définitions) de vos fonctions.

Exercice 1 Ecrivez une fonction `afficheTab` qui prend en paramètre un tableau de `char` et un entier représentant la taille de ce tableau, et affiche (à l'aide de printf) le contenu du tableau. La fonction ne retourne donc rien.
Pour tester cette fonction, créez un tableau de `char` dans le main contenant un mot et appelez votre fonction afin de l'afficher.
Pour créer rapidement un tableau de caractères en C, on peut utiliser le code suivant :
`char mot[] = "chateau";`

Exercice 2 Ecrivez une fonction `gagne` qui prend en paramètres deux mots (sous forme de tableaux de `char`) et un entier représentant leur taille (on supposera qu'ils sont de même taille), et qui retourne 1 si les mots sont identiques, et 0 sinon.
N'oubliez pas de tester votre fonction dans le main afin de vous assurer qu'elle fonctionne correctement.

Exercice 3 Ecrivez une fonction `existe` qui prend en paramètre un mot (sous forme d'un tableau de `char`), une lettre (sous forme d'une variable de type char) ainsi qu'un entier représentant la taille du mot, et qui retourne 1 si la lettre est présente dans le mot, et 0 sinon.
N'oubliez pas de tester votre fonction dans le main afin de vous assurer qu'elle fonctionne correctement.

Exercice 4 Ecrivez une fonction `miseAJour` qui prend en entrée deux mots (sous forme de tableaux de `char`), un entier représentant leur taille (on supposera qu'ils sont de même taille), ainsi qu'une lettre (sous forme d'une variable de type `char`). Cette fonction ne retourne rien, n'affiche rien, mais va remplacer dans le second mot toutes les occurences de la lettre dans le premier mot à la même place. Exemple :
– premier mot : `[ c h a t e a u ]`
– second mot : `[ - h - t - - u ]` (les tirets étants des caractères comme les autres)
– donc la taille : 7
– la lettre : a
la fonction doit modifier le second mot pour qu'il soit égal à `[ - h a t - a u ]`
N'oubliez pas de tester votre fonction dans le main afin de vous assurer qu'elle fonctionne correctement.

Exercice 5 Recopiez la fonction `lireChar` suivante :

```
1 char lireChar() {
2     char lettre[2];
3     printf("Votre proposition : ");
4     scanf("%s", lettre);
5     return lettre[0];
6 }
```

Exercice 6 Ecrivez enfin la fonction récursive `pendu` en remplissant le code à trous suivant. Les trous, représentés par des #####, peuvent être des instructions, conditions, appels de fonctions...etc. N'oubliez pas d'utilisez les fonctions que vous venez d'écrire !

```
int pendu(int nombreDeVies, char* motADeviner, char* reponse, int taille) {

    system("cls"); //cette instruction permet d'effacer le texte precedent dans la console

    if ( ##### ) {
        //cas ou il n'y a plus de vie

        printf("Vous n'avez plus de vie :-( \n");
        printf("Le mot a deviner etait : ");
        #####
        printf("\n");
        return 0;
    } else {
        if ( ##### ) {
            //cas ou le joueur a decouvert le mot en entier

            printf("Vous avez gagne, bravo !\n");
            return 0;
        } else {
            //cas ou il reste des vies et des lettres a decouvrir

            printf("Nombre de vies restantes : %d\n\n", ##### );
            printf("Votre mot courant : ");
            #####
            printf("\n\n");

            char nouvelleLettre = #####

            if (existe(#####, #####, #####) == 1) {
                //si la lettre saisie fait partie du mot

                miseAJour(#####, #####, #####, #####);
                return pendu(#####, #####, #####, #####);

            } else {
                //si la lettre saisie ne fait pas partie du mot

                return pendu(#####, #####, #####, #####);
            }
        }
    }
}
```

Exercice 7 Essayez enfin dans le main votre fonction `pendu` en l'appelant avec comme paramètres :
- `nombreDeVies` : une valeur, par exemple, 5.
- `motADeviner` : un tableau de `char` représentant un mot à deviner.
- `reponse` : un tableau de `char` de la même taille que `motADeviner` initialisé avec des tirets, et représentant le mot courant que le joueur est en train de deviner.
- `taille` : la taille de votre mot à deviner (= taille des tableaux `motADeviner` et `reponse`).

EXERCICE BONUS :

Exercice 8 Dans notre jeu, le mot à deviner est unique et fixé dans le code, ce n'est pas bien passionnant. Trouvez un moyen pour avoir un tableau de plusieurs mots, et tirez un mot aléatoirement à chaque exécution de votre programme.
Indication : pour générer un entier entre 0 (inclu) et 10 (exclu), on a le code suivant :

```
        srand (time (NULL));
        int id = rand() % 10;
```

En important la librairie `<time.h>`.