Correction TD chaines de caractères

Exercice d'application (cours):

Ecrire une fonction qui cherche les lettres d'un mot éparpillées dans une phrase, dans le même ordre.
Il s'agit de vérifier si les lettres d'un mot sont bien présentes dans une phrase, ce dans le même ordre que celui du mot.
Exemple :

phrase : le chat est gris et boit.
mot : lattis
Résultat : vrai

Solution

```
#include <stdio.h>
#include<conio.h>
#include <string.h>
#include <ctype.h>
#include <stdlib.h>
void lect_phrasemot(char *ph,char *ch)
{
        printf("donner une phrase\n");
        gets(ph);
        printf("donner un mot\n");
        scanf("%s",ch);
}

int trait(char *ph,char *ch)
{
   int i,j=0,c=0;
   char *p,*q,*p1;
   p=ph; // initialisation des pointeurs
   q=ch;
   p1=strchr(p,*q);
   q++;
   while(*p1!='\0')

   {
       p1=strchr(p1,*q);
```

```
        if(p1==NULL)
                return 0;
        q++;
    }
 return 1;
}

int main ()
{
char *mot;char *ph;
mot=(char*)malloc(20);
ph=(char*)malloc(50);
lect_phrasemot(ph,mot);
puts(mot);
puts(ph);
if (trait(ph,mot)==1)
printf("les lettres du mot ( %s ) existent dans la chaine ( %s )",mot,ph);
else
printf("les lettres du mot ( %s ) n'existent pas dans la chaine ( %s )",mot,ph);
free(mot);
free(ph);
getch();
return 0;

}
```

Exercice n°2:

On désire écrire un programme qui permet de conjuguer un verbe régulier du premier groupe, pour cela on demande d'écrire les fonctions suivantes :

1. La fonction **estpremier(char *verb)** qui saisit un verbe et renvoi 1 s'il s'agit d'un verbe du premier groupe, 0 sinon.
2. La fonction **conjugue(char *verv)** qui conjugue le verbe au présent avec tous les pronoms personnels (je, tu, il/elle ,nous, vous, ils/elles)
3. La fonction **main()** qui permet de tester les fonctions précédentes

   Exemple d'exécution le verbe : marcher

   je marche
   tu marches
   il marche
   nous marchons
   vous marchez
   ils marchent

NB : faite attention au cas des verbes comme habiter (**j'**habite) et manger(nous mang**eons**)

Solution

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
#include<string.h>
#include <ctype.h>
```

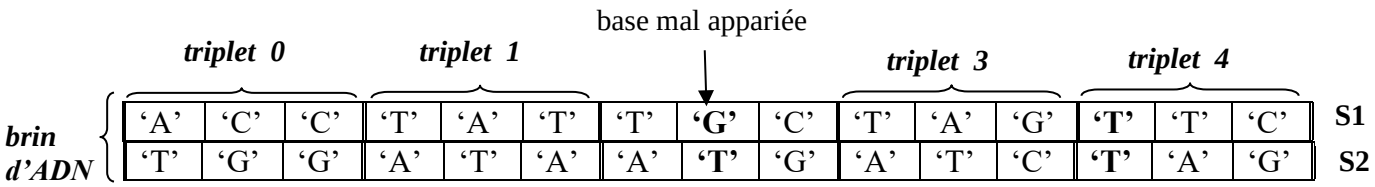
```
char *pron[]={"je","tu","il/elle","nous","vous","ils/elles"};
char *term[]={"e","es","e","ons","ez","ent"};
////////////////////////////////////////////////////////////////////////////////
// CONVERTIR UNE CHAINE EN MAJUSCULE
////////////////////////////////////////////////////////////////////////////////
void min_maj(char *s,char *ch)
{
        char *p;
        for(p=s;p<=s+strlen(s);p++) // on met <=  pour copier le '\0'
                ch[p-s]=toupper(*p);
}
////////////////////////////////////////////////////////////////////////////////
// TESTER SI LE VERBE EST DU PREMIER GROUPE
////////////////////////////////////////////////////////////////////////////////
int estpremier(char *ch)
{
char *p=ch;
char s[20];
min_maj(ch,s);
if(strcmp(s+strlen(s)-2,"ER") ==0 &&strcmp(s,"ALLER")!=0) // appel pour les versions 1 et 2 de
                                                          //la fonction min_maj
return 1;
return 0;
}
////////////////////////////////////////////////////////////////////
//      VERSION 1
////////////////////////////////////////////////////////////////////
/*int voyelle(char c)
{
if((toupper(c)=='A')||(toupper(c)=='E')||(toupper(c)=='Y')||(toupper(c)=='U')||
( toupper(c)=='I')||( toupper(c)=='O')||( toupper(c)=='H')) return 1;
else return 0;
}*/

////////////////////////////////////////////////////////////////////
//      VERSION 2
////////////////////////////////////////////////////////////////////
int voyelle(char c)
{
if(strchr("AEIYUOH", toupper(c))!=NULL)
return 1;
return 0;
}
////////////////////////////////////////////////////////////////////////////////
//      CONJUGUAISON
////////////////////////////////////////////////////////////////////////////////
void conjugue(char * ch)
{
int i;
char affi[30],racine[30];
strncpy(racine,ch,strlen(ch)-2);// récupérer la radicale du verbe
```

```
racine[strlen(ch)-2]='\0';

for(i=0;i<=6;i++)
{
        strcpy(affi,pron[i]);
        strcat(affi," ");
        strcat(affi, racine);
        strcat(affi,term[i]);
        puts(affi);
}
}

////////////////////////////////////////////////////////////////////////////////////////////////////
//            PROGRAMME PRINCIPAL
////////////////////////////////////////////////////////////////////////////////////////////////////
int main()
{       int x;
        char ch[40];
        printf("\n donner un verbe a conjuguer: \n");
        gets(ch);
        x=estpremier(ch);
        if(x==1)
        {
         if(voyelle(ch[0])==1 ) // cas du verbe qui commence par une voyelle ou la lettre 'h'
        {
          pron[0]=(char *)malloc(3);  // il faut allouer  l'espace nécessaire pour la nouvelle chaine
          strcpy(pron[0],"j'");
        }
        if(ch[strlen(ch)-3]=='g') // cas des verbes qui se terminent par ger
        {
           term[3]=(char *)malloc(5);  //  il faut allouer l'espace nécessaire pour la nouvelle chaine
           strcpy(term[3], "eons");
        }

        printf("\n la conjugaison du verbe %s  au present de l'indicatif \n",ch);
        conjugue(ch);
        return 0;
}
else
        printf("\n le  verbe n'est pas du premier groupe!");

return 0;
}
```

Exercice n°3: *Manipulation des chaines de caractères*

L'**ADN** s'organise en deux **brins para**llèles constitués de **triplets** des quatre bases **A**, **T, C** et **G**. Les bases **A** et **T** (respectivement, les bases **C** et **G)** forment des paires complémentaires : les bases d'une paire se font face.

Exemple :



1. Écrire une fonction **Nbmalappariee()** qui compte et retourne le nombre de bases mal appariées entre deux brins **ADN** donnés **S1** et **S2** (et renvoyant -1 si **S1** et **S2** ne sont pas de même longueur). *Dans l'exemple ci-dessous nous avons 2 bases mal appariées*
2. Un brin d'**ARN** est constitué d'un seul brin obtenu comme *copie partielle* d'un brin d'ADN mais où ***la base U remplace la base T***.
   Écrire une fonction **Transcription()** qui construit le brin **ARN** à partir d'un brin **ADN** donné **S**2 et des indices *prem* et *dern* des premier et dernier ***triplets*** copiés.

Exemple :

Copie à partir du brin de S2 avec prem=1 et dern=4

brin d'ARN

'A'	'U'	'A'	'A'	'U'	'G'	'A'	'U'	'C'	'U'	'A'	'G'

3. Écrire une fonction **PartieCodante()** qui extrait la partie codante d'un brin **ARN**, nommée **PC** si c'est possible et renvoie NULL sinon.

La partie codante d'un brin **ARN** est strictement comprise entre un triplet de début "**AUG**" et un triplet de fin ("**UAA**" ou "**UAG**" ou "**UGA**"), triplets les plus à droite.

Exemple : *la partie codante du brin **d'ARN** suivant* :

'A'	'U'	'A'	'A'	'U'	'G'	'A'	'U'	'C'	'U'	'A'	'G'

*est donné par la chaine **PC** suivante :*

'A'	'U'	'G'	'A'	'U'	'C'	'U'	'A'	'G'

4. Écrire une fonction **NbTryptophane()** comptant le nombre de triplets "**UGG**" codant l'acide aminé tryptophane dans la partie codante d'un brin **ARN** donné.

Solution

```
#include<stdio.h>
#include<conio.h>
#include<string.h>

int nbapp(char *s0,char *s1)
{
char *p,*p1;
int nb=0;
if(strlen(s0)!=strlen(s1))
        return -1;
else
```

```
        for(p=s0,p1=s1;p<s0+strlen(s0);p++,p1++)
                if( (*p=='A' && *p1!='T')||(*p=='T' && *p1!='A')||(*p=='C' &&
                *p1!='G')||(*p=='G' && *p1!='C'))
                        nb++;
        return nb;
}

void transcription( char *s,char *arn,int prem,int dern)
{
 char *p,*q;
for(p=s+prem*3,q=arn;p<s+(dern+1)*3;p++,q++)
        if(*p=='T')
                *q='U';
        else
                *q=*p;
*q='\0';
}

void partiecodante(char *arn,char *pc)
{
char *p,*q;
p=arn;
q=arn+strlen(arn)-3;
// recherche du triplet “AUG”

while((p<arn+strlen(arn)-3)&&(strncmp(p,"AUG",3)!=0) )
        p=p+3;

// recherche du triplet “UAG”

while(q>=arn+3 && strncmp(q,"UAG",3)!=0)
        q=q-3;
if(p<q)
{
        strncpy(pc,arn+(p-arn),q-p+3);
        *(pc+(q-p+3))='\0';
}
}

int NbTryptophane(char *s)
{
        char *p;
        int nb=0;
        puts(s);
        for(p=s;p<s+strlen(s)-3;p=p+3)
                if(strncmp(p,"UGG",3)==0)
                        nb++;
        return nb;
 }
```

```
int main()
{
  char arn[20],pc[20];
  char adn[27]="-";
  char s[]="TGGATAATGATCTAGAAC";
  printf("\n\t\t LES DEUX BRINS D'ADN \n");
  printf("\t\t\t\t\t");
  puts (adn);
  printf("\t\t\t\t\t");
  puts(s);
  printf("\n\t\t LE NOMBRE DE BASES MAL APPARIEES=  %d\n\n",nbapp(adn,s));
  transcription(s,arn,0,4);

   printf("\n\t\t LE BRIN D'ARN :    ");
   puts(arn);
   printf("\n\n\t\t LA PARTIE CODANTE DU BRIN D'ARN \" %s \"  EST DONNEE PAR LA
   CHAINE :",arn);
   partiecodante(arn,pc);
   puts (pc);
   printf("\n\t\t LE NOMBRE DE TRYPTOPHANE = %d  ", NbTryptophane(arn));
   getch();
   return 0;
}
```