SOLUTIONS FOR EXEMPLARY TASKS
illustrating the writing of **text-processing functions**

TASK_1 Write a function **DELETE_LOWERCASE** , which removes all lowercase letters, from the string of characters (given as input to this function).

TASK_2 Write the function **COMPARE_LOWER_UPPER** checking whether in the text (given as an argument) there are more lowercase or uppercase letters. Depending on the results of the comparison, the function should return a number: positive, negative or zero.

TASK_3 Write the **DELETE_DOUBLE_SPACES** function that deletes all "*multiple spaces*" , from the text given as a parameter.

TASK_4 Write the function **DELETE_FIRST_MINUS** removing (by shortening the text) the first occurrence of the minus character '–' , from the string of characters passed as input to this function.

TASK_5 Write a function **ADD_5_SPACES** which adds 5 spaces to the beginning of the text, passed as an input parameter of this function.

TASK_6 Write the function **DELETE_COMMENT** which deletes (from the text passed as a parameter) the second part of the string, starting with the comment "//" sequence.

TASK_7 Write a function that **HAS_TXT_EXTENSION** checking whether the text (passed as a parameter of this function) ends with the sequence "**.txt**" (just like the names of text files).

---

1. Write a function, **DELETE_LOWERCASE** , which removes all lowercase letters, from the string of characters (given as input to this function).

```
char* DELETE_LOWERCASE( char t[])
{
   int i = 0;
   while (t [i]! = '\ 0')
   if (t [i]> = 'a' && t [i] <= 'z')
     strcpy ( &t[i], &t[i + 1]);
   else
     i++;
   return t;
}
```

```
// the second (faster) version of the same function
char* DELETE_LOWERCASE_2( char t[])
{
   int i=0, j=0;
   while( t[i] != '\0' )
     if( t[i]>='a' && t[i]<='z' )
       i++;
     else
       t[j++] = t[i++];
   t[j] = '\0';
   return t;
}
```

2. Write the function **COMPARE_LOWER_UPPER** checking whether in the text (given as an argument) there are more lowercase or uppercase letters. Depending on the results of the comparison, the function should return a number: positive, negative or zero.

```
#include <stdio.h>

int COMPARE_LOWER_UPPER( char t[])
{
  int counter_lower=0, counter_upper=0;
  for( int i=0; t[i] != '\0'; i++)
    {
      if( t[i]>='a' && t[i]<='z')
        counter_lower++;
      if( t[i]>='A' && t[i]<='Z')
        counter_upper++;
    }
  return counter_lower - counter_upper;
}



int main () // example program, using the above function,
{           // with text loaded from the keyboard
  char text[100];
  printf( "Enter a text to check:" );
  fgets( text, 100, stdin );

  int result = COMPARE_LOWER_UPPER(text);

  if( result>0 )
    printf( "more lowercase letters" );
  else if( result<0 )
    printf( "more uppercase letters" );
  else
    printf( "there are as many lowercase as uppercase" );
  return 0;
}
```

3. Write the **DELETE_DOUBLE_SPACES** function that deletes all "*multiple spaces*" , from the text given as a parameter.

```
char* DELETE_DOUBLE_SPACES( char t[] )
{
  if( t[0] == '\0' || t[1] == '\0')
    return t;
  int i = 1;
  while( t[i] != '\0' )
    if( t[i-1]==' ' && t[i]==' ' )
      strcpy( &t[i], &t[i+1] );
    else
      i++;
  return t;
}
```

4. Write the function **DELETE_FIRST_MINUS**
removing (by shortening the text) the first occurrence of the minus character '–',
from the string of characters passed as input to this function.

```
#include <string.h>

void DELETE_FIRST_MINUS( char t[] )
{
  int position=0;

  // search for first '-' character
  while( t[position]!=0 && t[position]!='-' )
    ++position;

  // delete the character (if found)
  if( t[position]=='-' )
    strcpy( &t[position], &t[position+1] );
}
```

5. Write a function **ADD_5_SPACES** which adds 5 spaces to the beginning of the text,
passed as an input parameter of this function.

```
#include <string.h>

void ADD_5_SPACES( char t[] )
{
  // shift all letters, 5 positions to the right
  memmove( t+5, t, strlen(t)+1 );

  // fill the beginning with spaces
  for( int i=0; i<5; i++ )
    t[i]=32;
}
```

6. Write the function **DELETE_COMMENT** which deletes (from the text passed as a parameter)
the second part of the string, starting with the comment "//" sequence.

```
#include <string.h>

void DELETE_COMMENT ( char t[] )
{
  // find the position of the comment characters
  char* wsk;
  wsk = strstr( t, "//" );

  // trimming the right fragment, starting from the found position
  if( wsk!=NULL )
    *wsk=0;
}
```

7. Write a function that **HAS_TXT_EXTENSION** checking whether the text (passed as a parameter of this function)
ends with the sequence "**.txt**" (just like the names of text files).

```
#include <string.h>

int HAS_TXT_EXTENSION( char name[] )
{
  int length = strlen (name);

  // immediate return, if the name has less than 4 letters
  if (length <4)
    return 0; // return false

  // check, if the last four letters are ".txt"
  if( strcmp( name+length-4, ".txt" )==0 )
    return 1; // return true
  else
    return 0;
}
```