Ejercicio 1 (2.5 puntos)

Se tiene una cadena que contiene números decimales separados por coma. Se requiere implementar la función void **normalizar** que reciba como único parámetro un string y lo normalice dejando todos los números con los decimales truncados a 2.
Se asegura que el string está bien formado:

1. Sólo contiene números decimales separados por comas
2. Los números no tienen signo
3. Todos los números tienen al menos un dígito en la parte entera y un dígito en la parte decimal, separados por un punto.

Ejemplos

- si el string original es "`12.33333,23.44444,1.0054,5.003,7.0`" queda "`12.33,23.44,1.00,5.00,7.0`"
- si el string original es "`12.33,3.4,1.00,5.0,7.1`" queda igual

Posibles soluciones (respuestas dadas por alumnos)

```
void normalizar(char * s) {
   int i, dim=0, count =0;
   char anterior=0;
   for (i=0 ; s[i] != '\0' ; i++) {
      if( anterior == ',' || i == 0 )
         while(s[ i ]  !=  '.' )
            s[dim++] = s[i++];
      if( ! isdigit( s[i] )
         s[dim++] = s[ i ];
      else if( anterior == '.' )
         count=2;
      if(count > 0) {
         if(isdigit( s[ i ] ) {
            s[dim++] = s[ i ];
         }
         count--;
      }
      anterior = s[ i ];
   }
 s[dim] = 0;
}
```

```
void normalizar (char str[]){
  int newDim = 0, i = 0;
  while (str[i] ){
     while (isdigit(str[i])){
        str[newDim++] = str[i++];  //mientras no llegue al '.' copio los enteros
     }
    //como me aseguran que hay al menos un digito en la parte entera y uno en la parte decimal, sé que el while de arriba cortó porque es un '.'
      str[newDim++] = str[i++];      //copio el '.' y paso al siguiente
      int copiar = 2; //2 decimales a copiar
      while ( copiar > 0 && str[i] != ',' && str[i]){     //mientras no haya copiado 2 numeros, no haya llegado a la coma y no haya llegado al final
          str[newDim++] = str[i++];
          copiar--;
       }
       while (str[i] && str[i] != ',' ){     //mientras no haya llegado al final y no sea una ',' avanza
           i++;
       }
      if (str[i] ){      //si no llegó al final , el while de arriba cortó porque es ','
        str[newDim++] = str[i++]; //copio la ',' y paso al siguiente
      }
    //si llegó al final, va a cortar en el primer while
  }
  str[newDim] = 0;
}
```

Ejercicio 2 (3 puntos)

Escribir una función que reciba dos matrices cuadradas de enteros, de dimensión DIM (DIM es una constante simbólica ya definida) y retorne:

- 1 si cada elemento de la fila n de la primera matriz está en la columna n de la segunda matriz, para todo n entre 0 y DIM-1
- 2 si cada elemento de la columna n de la primera matriz está en la fila n de la segunda matriz, para todo n entre 0 y DIM-1
- En caso de cumplir ambas condiciones puede retornar 1 ó 2
- 0 si no se cumple ninguna de las dos condiciones

Las matrices pueden tener elementos repetidos y no hay ningún tipo de orden,
Ejemplo: asumiendo que DIM es 4

Si recibe las siguientes matrices debe retornar 1

1	3	1	2
5	5	6	6
2	3	4	5
1	1	1	2

1	6	5	2
2	6	3	2
3	6	2	1
3	5	4	1

Si recibe las siguientes matrices debe retornar 2

1	3	1	2
5	5	6	6
2	3	4	5
1	1	1	2

1	2	5	2
1	3	3	5
1	4	4	6
2	5	6	6

Posibles respuestas (dadas por alumnos)

```
static int perteneceFilaCol(int fila[], int m[][DIM], int j){
    int flag=1;
    for (int i=0; i < DIM && flag; i++){
        flag=0;
        for (int k=0; k < DIM && !flag; k++){
            if (fila[i] == m[k][j])
                flag=1;
        }
    }
    return flag;
}
static int control(int m1[][DIM], int m2[][DIM]){
    int flag=1;
    for (int i=0; i < DIM && flag!=0; i++){      //corta si llega al final, o
encuentra 1 fila en la matriz que no cumplen la condicion
        flag=perteneceFilaCol(m1[i], m2, i);
    }
    return flag;
}
int belongs(int m1[][DIM], int m2[][DIM]){
    int a=1, b=1;

    a=control(m1, m2);
    if (a!=0){       //el ejercico indica que es indistino el retorno si cumple
ambas condiciones
        b=control(m2, m1)  // de esta manera, con cumplir una, no recorre
de nuevo las matrices
    }
    if (a)
        return 1;
    else return (b)?2:0;
}
```

```
int casos(const int m1[], int m2){
int i;
int encontre = 0;
 for(i = 0; i < DIM && !encontre; i++){
  if(m1[i] == m2)
   encontre = 1;
 } // for i
return encontre;
}
int matrices(const int m1[][DIM], const int m2[][DIM] ) {
 int i, j;
 int caso1 = 1;
 int caso2 = 1;
 int resp = 1;
 for (i =0; i < DIM && caso1 ; i++){
  for(j = 0; j < DIM && caso1 && resp; j++){
   resp = casos(m1[i], m2[j][i]);
  } // for j
   if (!resp )
     caso1 = 0; // entonces no devuelvo 1
 } // for i
if(!caso1){
 for (i =0; i < DIM && caso2 ; i++){
  resp = 1;
  for(j = 0; j < DIM && caso2 && resp ; j++){
   resp = casos(m2[i], m1[j][i]);
  } // for j
   if (!resp )
     caso2 = 0; // entonces no devuelvo 2
 } // for i
 if(caso2)
   return 2;
 } // if de caso1
else
  return 1;
return 0;
}
```

Ejercicio 3 (1.5 puntos)
Escribir la función **endsWith** que reciba solamente dos strings y retorne 1 si el final del primer string es igual al segundo string, y cero si no.

Para los siguientes ejemplos debe retornar 1

```
endsWith("hola","la");
endsWith("hola","hola");
endsWith("hola","");
endsWith("","");
endsWith("lalala","la");
```

Para los siguientes ejemplos debe retornar 0

```
endsWith("hola"," hola");
endsWith("la","lala");
endsWith("","x");
endsWith("lasollafado","la");
endsWith("laa","la");
endsWith("hola","LA");
```

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

// strend EJ1-A TP7

int endsWith(const char *s, const char *t) {
	int ls, lt;

	/* Se calcula la longitud de las cadenas */
	ls = strlen(s);
	lt = strlen(t);

	if (lt > ls) /* Si la cadena t es mas larga, es falso */
		return 0;

	for (ls--, lt--; lt >= 0 && s[ls] == t[lt]; ls--, lt--)
		;

	return (lt == -1);
}

/* Version mas simple */
int endsWith2(const char *s, const char *t) {
	int ls, lt;

	/* Se calcula la longitud de las cadenas */
	ls = strlen(s);
	lt = strlen(t);
```

```
	if (lt > ls) /* Si la cadena t es mas larga, es falso */
		return 0;
	return  strcmp(s + ls - lt, t) == 0;
	/* La funcion tambien podria ser unicamente la siguiente linea
		return strcmp(s + strlen(s) - strlen(t), t) == 0;
	** pero podria dar error de acceso a memoria invalida si lt > ls
	*/
}

int main(void) {
	assert(endsWith("hola","la") == 1);
	assert(endsWith("hola","hola") == 1);
	assert(endsWith("hola","") == 1);
	assert(endsWith("","") == 1);
	assert(endsWith("lalala","la") == 1);

	assert(endsWith("hola"," hola") == 0);
	assert(endsWith("la","lala") == 0);
	assert(endsWith("","x") == 0);
	assert(endsWith("lasollafado","la") == 0);
	assert(endsWith("laa","la") == 0);
	assert(endsWith("hola","LA") == 0);
}
```

Ejercicio 4 (3 puntos)

Escribir una función **`analize`** que reciba un string de nombre **`text`** (se espera que sea muy extenso) y un vector de chars de nombre **`chars`** que tiene al menos 256 posiciones reservadas pero no inicializadas (contienen basura). La función debe dejar en **`chars`** los distintos caracteres que aparecen en **`text`** , ordenados ascendentemente por valor ASCII y sin repetir. El vector **`chars`** debe quedar *null terminated*.

Como siempre, se asegura que el string contiene únicamente caracteres US-ASCII (entre 1 y 255).

El siguiente programa

```
#define CHARS_DIM 256

int main(void) {
 char chars[CHARS_DIM];
 analize("El zorro mete la cola, Pero no la pata!", chars);
 printf("%s\n", chars);
 return 0;
}
```

imprimirá lo siguiente (sin las comillas)

```
" !,EPacelmnoprtz"
```

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

#define CHARS_DIM 256

static void eliminaCeros(char [], size_t);
void analize(const char [], char[]);

void analize(const char text[], char chars[]) {
   // Opcional: Contabilizar los caracteres distintos
   int dim = 0;
   // Inicializo el vector resultado chars
   // Otra opciÃ³n puede ser declarar un arreglo auxiliar inicializado en cero,
   // por ejemplo: char esta[CHARS_DIM] = {0};
   for(int i = 0; i < CHARS_DIM; i++) {
      chars[i] = 0;
```

```
    }
    // Copio cada caracter de text en chars usando el caracter como Ãndice
    for(int i = 0; text[i]; i++) {
        int c = text[i];
        if(chars[c] == 0) {
            chars[c] = c;
            dim++;
        }
    }
    // Elimino las posiciones vacÃas de chars
    eliminaCeros(chars, dim);
}

static void eliminaCeros(char s[], size_t dim) {
    int i, j;
    for(i = j = 0; i < CHARS_DIM && j < dim; i++) { // Opcional: j < dim
        if(s[i]) {
            s[j++] = s[i];
        }
    }
    s[j] = 0;
}

int main(void) {
    char chars[CHARS_DIM];
    analize("El zorro mete la cola, Pero no la pata!", chars);
    printf("%s\n", chars);
    assert(strcmp(chars, " !,EPacelmnoprtz") == 0);
    return 0;
}
```