UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

R1051
Informática I

Primer Parcial
04/Jul/2016

Apellido, Nombre:_____

Legajo N°:_____

Condiciones

El exámen consta de una parte práctica con dos ejercicios y una parte teórica.
Para aprobar es necesario tener bien realizado, sin errores de compilación y corriendo las funciones uno de los dos ejercicios de la parte práctica, y no tener errores conceptuales en la parte teórica.
Está permitido usar código realizado previamente, funciones de la librería estándar, apuntes y bibliografía.

Armar un árbol de directorio con el siguiente formato:

```
apellido_nombre/
      ejercicio1/
            trim.c
            trim.h
            test_trim.c
            Makefile
      ejercicio2/
            msimetrica.c
            msimetrica.h
            test_simetrica.c
            Makefile
      teorico.txt
```

Parte Práctica

1.a. Implementar la función `ltrim` *"left trim"* que quita los espacios en blanco a la izquierda de un string. Su prototipo es: **`char *ltrim(char *s);`**
`ltrim` recibe el puntero al string del que se quieren quitar los espacios y devuelve un puntero al string sin espacios.
La operación se debe realizar "in-place", o sea, sin necesidad de mover o copiar los datos.
Ej: recibe " hola ", devuelve "hola "

1.b. Implementar la función `rtrim` *"right trim"* que quita los espacios en blanco a la derecha de un string. Su prototipo es: **`char *rtrim(char *s);`**
`rtrim` recibe el puntero al string de que se quieren quitar los espacios y devuelve un puntero al string sin espacios.
La operación se debe realizar "in-place", o sea, sin necesidad de mover o copiar los datos.
Ej: recibe " hola ", devuelve " hola"

1.c. Implementar la función `trim` que utiliza las dos funciones antes desarrolladas y quita los espacios en blanco tanto a izquierda como a derecha. Su prototipo es **`char *trim(char *s);`**
La operación se debe realizar "in-place", o sea, sin necesidad de mover o copiar los datos.
Ej: recibe " hola ", devuelve "hola"

Punto bonus. Implementar un programa `test_trim` de pruebas que utiliza la función `trim` para convertir todos los strings recibidos por línea de comandos excepto el nombre del programa y los imprima en pantalla, antes de *trimmearlo* y después de *trimmearlo*. Se deberán poner las funciones en un archivo aparte, escribir el archivo de headers y el makefile para generar el ejecutable. Recordar que para pasar por línea de comandos un string que posee espacios, hay que pasarlo entre comillas.
xej: **`./test_trim "  hola  " "chau  " "      hasta luego"`**
imprime en pantalla:

```
'  hola  ' >> 'hola'
'chau  ' >> 'chau'
'      hasta luego' >> 'hasta luego'
```

2.a. Implementar la función `check_simetrica` que verifica si una matriz cuadrada de NxN es simétrica. Recordar que una matriz es simétrica sí y sólo sí, cada elemento de la matriz $a_{ij}=a_{ji}$ donde *i* es la fila y *j* es la columna. Su prototipo es **`int check_simetrica(int *m, int n);`**
`check_simetrica` recibe un puntero a la matriz `m` y la dimensión de la matriz (cantidad de filas/columnas) `n`, y devuelve `1` si es simétrica ó `0` si no lo es.

2.b. Implementar un programa de pruebas para `check_simetrica` llamado `test_simetrica`. Las matrices de prueba se pueden inicializar en el código, tipo: **`int matriz[N][N] = {{….}, {….}, ….};`**

Punto bonus. Implementar dos funciones para llenar las matrices de prueba, una que la llene simétrica y otra que no. Utilizar las funciones de generación de números random vistas en clase. Se deberán poner las funciones en un archivo aparte, escribir el archivo de headers y el makefile para generar el ejecutable.

Parte teórica
Considere el siguiente bloque de código y complete la lado de cada `printf` la salida en pantalla "justificando" la respuesta. Asuma que las direcciones asignadas por el sistema operativo son para: `b=0x10000100, p=0x10000200, a=0x10000300, x=0x10000400 y z=0x10000500`

```
#include <stdio.h>      // para printf
#include <string.h>     // para strcpy
#include <ctype.h>      // para toupper

int main(void){
    char b[100], *p;
    char a[] = {'a', 'b', 'c', 'd'};
    char *x, *z;

    x = a;
    z = &a[2];
    strcpy(b, "hola");
    printf("%d\n", *a);                  __________
    printf("%c\n", *(z-2));              __________
    printf("%ld\n", x-z);                __________
    printf("%d\n", (*(x+3))-(*z+1));     __________
    printf("%s\n", b);                   __________
    b[2] = toupper(b[2]);
    printf("%s\n", b);                   __________
    printf("%s\n", strcpy(p, b));        __________

    return 0;
}
```