TRABAJO PRÁCTICO N° 6
REPASO EXAMEN

Ejercicios de repaso para el examen con los temas vistos hasta la fecha.
No se debe entregar, es solo para practicar.

1. Escribir una función llamada `ordenalfa` que determina el orden alfabético de dos palabras. El prototipo es: `int ordenalfa (char* str1, char* str2);`
   La función recibe dos *strings* cuyo contenido son palabras (solamente letras), y debe devolver `-1` si `str1` está antes que `str2`, `1` si `str1` está después que `str2` o `0` si `str1` y `str2` son iguales. Debe ignorar mayúsculas y minúsculas (i.e: debe ser *case insensitive*) y no se pueden usar funciones de la librería `string`.

2. Escribir un programa que imprima `"Hola Mundo"` o `"Hello World"` dependiendo del valor de la constante `IDIOMA`, mediante el uso de instrucciones de preprocesador y directivas condicionales de compilación.

3. Escribir una función llamada `rombo` que imprime en pantalla un rombo relleno con asteriscos de lado `n`. La función recibe el parámetro `n` y no devuelve nada. Debe funcionar para todo $1 \leq n \leq 25$. No se permite utilizar arreglos. Se debe validar `n`.
   Por ejemplo:

   `rombo(3)` imprime:
   ```
     *
    ***
   *****
    ***
     *
   ```

   `rombo(4)` imprime:
   ```
      *
     ***
    *****
   *******
    *****
     ***
      *
   ```

   `rombo(1)` imprime:
   ```
   *
   ```

4. En una terminal de Linux se desea:
   a. Compilar un programa guardado en el archivo `ejercicio.c`, con el compilador mostrando todos los *warnings*, creando el archivo ejecutable `programa` y generando la información para poder "debuggearlo".
   b. Ejecutar el archivo compilado.
   c. "Debuggear" el programa.

   Escribir los comandos para realizar dichas acciones.

5. Se tiene la siguiente macro para determinar si un caracter es letra del alfabeto inglés o no, devolviendo 1 o 0 respectivamente:
   ```
   #define ISLETTER(ch)  c > 'A' & c < 'Z' | c > 'a' & c < 'z'
   ```
   Sin embargo, la misma contiene errores y funciona mal. Corregirla para que funcione correctamente en todos los casos.

6. Escribir la función `CamelCase` que recibe un *string* con texto en inglés y lo modifica para que la primera letra de cada palabra quede en mayúscula y el resto en minúscula. El texto es en idioma inglés (no tiene `ñ` ni tildes), es un texto válido y puede tener mezcladas mayúsculas y minúsculas. El prototipo es: `void CamelCase(char* str);`
   Por ejemplo, si la entrada es:
   `"Will I paSS the EXAM? I hOPe so. i'VE stuDIEd a lot... NOT!"`
   la función debe editarlo para que quede:
   `"Will I Pass The Exam? I Hope So. I'Ve Studied A Lot... Not!"`

   Se pueden utilizar las siguientes funciones (suponer que ya están escritas):
   - `char toUpper (char)`: Si recibe una letra minúscula devuelve esa letra en mayúscula, sino devuelve el mismo caracter.
   - `char toLower (char)`: Si recibe una letra mayúscula devuelve esa letra en minúscula, sino devuelve el mismo caracter.
   - `int isUpper (char)`: Devuelve si un caracter es una letra mayúscula.
   - `int isLower (char)`: Devuelve si un caracter es una letra minúscula.

7. Escribir una función que determine si ocurrió una pierna de póker.
   Una mano tiene 5 cartas y cada carta se identifican con un valor numérico (de 2 a 10, J = 11, Q = 12, K = 13 y AS = 1) y con un palo (♠ = 100, ♥ = 200, ♦ = 300 y ♣ = 400). La carta se compone de sumar el valor con el palo (211 es J♥).
   Una pierna ocurre cuando hay 3 cartas del mismo valor y distinto palo.
   El prototipo debe ser: `int pierna_poker (const int cartas[]);`
   La función debe devolver 0 si no hubo pierna, o si hubo el valor (sin palo) de las cartas de la pierna. No es necesario validar las cartas ni determinar si ocurrió otro juego mayor.

8. Escribir una función que parte un *string* en dos, en la primera ocurrencia de un separador. El prototipo es: `char* strSplit(char* str, char separador);`
   El primer parámetro es el *string* a partir y el segundo es un `char` con el caracter que considera separador. Devuelve un puntero al comienzo del segundo *string*. Si no puede partir el *string*, devuelve un puntero al terminador.
   Por ejemplo, si se invoca `nstr = strSplit(str, c);` donde:
   - `str` es `"Hola paisano"` y `c` es `' '` → la función reemplaza el espacio (`' '`) por un terminador y devuelve un puntero a la `'p'`.
   - `str` es
   - `"Palombo;43216;9,58"` y `c` es `';'` → la función reemplaza el primer punto y coma (`';'`) por un terminador y devuelve un puntero al `'4'`.
   - `str` es `"Cuidado"` y `c` es `';'` → la función no parte al *string* y devuelve un puntero al terminador.

9. Escribir una función que transponga una matriz de `N`X`N`.
   El prototipo debe ser: `void transponer (double mat[N][N])`, siendo `N` una constante ya definida. El resultado debe devolverse modificando la matriz original.
   Tip: La traspuesta de una matriz se obtiene reflejando los elementos a lo largo de su diagonal.

10. Indicar la salida del siguiente programa, justificando su respuesta:

```
#include <stdio.h>

char text[] = "Washing your car to make it rain doesn't work.";
void myprint(char *p);

int main(void)
{
    char *p = text + 20;

    myprint(p+5);
    myprint(&p[16]);
    myprint(p+30);
    myprint(&text[13]);
    myprint(text+46);

    return 0;
}

void myprint(char *p)
{
    while(*p != ' ' && *p)
        putchar(*p++);
    putchar('\n');
}
```

11. Se tiene una función para leer un renglón del *stdin* (tanto teclado como archivo), colocando el renglón leído en `str` (en formato *string* y sin incluir el ENTER) y devolviendo el largo del *string*. Sin embargo, el programa no funciona.
Mencionar los errores y re-escribir el programa corregido.

```
#include <stdio.h>

int getline (char* str)
{
    int c, len;

    while (c != '\n') // Mientras que no termine el renglón
    {
        c = getchar(); // Leo nuevo caracter
        str[++len] = c; // Guardo caracter en str y
                        // actualizo contador de caracteres
    }

    return len; // Devuelvo largo del string
}
```