UNIVERSIDAD DE CARABOBO

Facultad Experimental de Ciencias y Tecnología

Departamento de Computación

Fundamentos de Programación

Cod. Asignatura: TAO207 **Modelo:** 1

Taller 2

Normas de Entrega:

- El nombre de cada archivo **.c**, archivo de entrada y archivo de salida debe coincidir con lo indicado por cada ejercicio.
- Debe colocar la siguiente información en la cabecera de cada archivo fuente por medio de un comentario de bloque: Fecha, Nombre, Apellido, Cédula, Número Sección y Número del Modelo del Parcial.
- La entrega de los programas se deberá realizar por medio de un archivo comprimido (**.zip**), el cual, debe contener una sub-directorio por cada ejercicio. Cada sub-carpeta deberá tener el archivo fuente (**.c**) y la entrada del programa correspondiente.
- No adjunte los archivos de salida del programa, ni adjunte los ejecutables generados por el compilador.
- Puede utilizar las librerías del estándar de C.
- Recuerde que solamente puede utilizar elementos del lenguaje C vistos en clase, tales como: condicionales, ciclos, archivos y acciones nominadas.
- No se evaluará su entrega si emplea arreglos o estructuras (temas que no han sido cubiertos en clase todavía).
- Procure revisar el documento en su totalidad. El mismo contiene de 2 ejercicios.

No seguir las indicaciones antes mencionadas repercutirá en la nota final de su entrega.

Ejercicio 1. Mensajes Bovinos (10 ptos)

Cowsay, es una utilidad GNU que fue originalmente desarrollada por Tony Moroe mediante el uso del lenguaje de programación Perl, lenguaje que se caracteriza por brindar una gran flexibilidad al programador dado que permite representar una misma acción de numerosas formas diferentes.

Esta peculiar utilidad le permite al usuario transformar un texto arbitrario en un ASCII-art de un personaje diciendo aquél mensaje que fue proporcionado en un globo de texto, en el cual, el personaje que aparece en la obra de arte le da nombre al programa, y no es más y nada menos que una vaca.

Este programa le ha otorgado alegría a los desarrolladores desde 1999 y comúnmente se usa junto a la utilidad *fortune*, para recibir sabios mensajes de una vaca (¡imagine que ocurriría si fuesen dos vacas!).

Dada la importancia del programa, se ha creado un mirror (servidor de respaldo) del mismo en la conocida página para almacenar código fuente, Github (https://github.com/cowsay-org/cowsay), y se encuentra actualmente disponible a todo público.

Tomando en cuenta la relevancia histórica de cowsay, su equipo ha decidido realizar un homenaje al planear una reimplementación del mismo utilizando el lenguaje C y usted, ha recibido el honor de ser el desarollador encargado de la elaboración del programa.

Su tarea consiste en crear un programa llamado **01-cowsay.c** que dado un archivo de entrada, genere la siguiente salida:

Ejemplo de entrada y salida

Entrada: 01-mensaje.txt

```
Texto
ABC
def
*abc defghijkl 0-             limit
12345

        There's a risk, an it's of rain
                     it's cinema
```

Salida: 01-salida.txt

```
****************************************
*                                      *
*                                      *
* Texto                                *
* ABC                                  *
* def                                  *
* *abc defghijkl 0-             limit  *
* 12345                                *
*                                      *
*        There's a risk, an it's of rain *
*                     it's cinema      *
*                                      *
****************************************
 *
  :
^__^
(oo)\_______
(__)\       )\/\
    ||----w |
    ||     ||
```

(ASCII-art original de Tony Monroe, 1999)

Nota: Considere que el archivo puede estar vacío o incluso, puede tener líneas muy extensas (más de 65000 símbolos).

Ejercicio 2. Ajedrez 6x6 (10 ptos)

Sea un tablero de Ajedrez reducido de 6 x 6 casillas, en el cual solo hay una pieza a la vez, puede ser un una Torre (T) o un Caballo (C). Tal situación se representa en un archivo de entrada llamado **02-tableroin.txt** de la siguiente manera:

	1	2	3	4	5	6
1	*	*	*	*	*	*
2	*	*	*	*	*	*
3	*	*	*	*	*	*
4	*	*	*	C	*	*
5	*	*	*	*	*	*
6	*	*	*	*	*	*

Entrada: 02-tableroin.txt

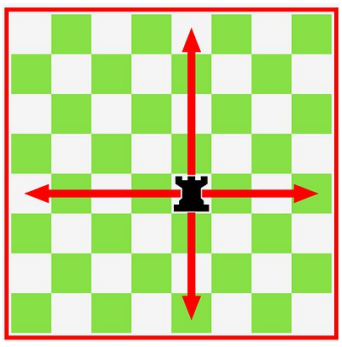
```
******
******
******
***C**
******
******
```

Un asterisco representa una celda vacía, y una letra (T, C) una celda ocupada por una Torre o un Caballo. Solo debe ir las celdas, la numeración de las filas y columnas es para referencia.

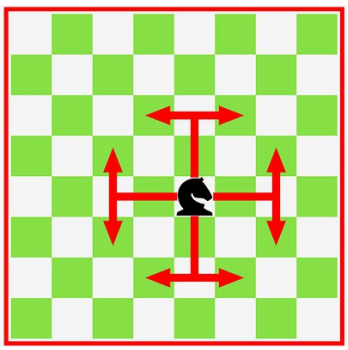
Codifique un algoritmo que lea ese archivo y determine la celda donde se encuentra la pieza (fila y columna) e imprima por pantalla esa información; luego imprima también una lista de los posibles movimientos (pares de filas y columnas) que puede realizar la pieza de acuerdo a la ubicación encontrada, es decir aquellas que estén dentro del tablero, de modo que el usuario puede elegir uno de esos posibles movimientos.

Una vez elegida la nueva posición de la pieza, se debe imprimir el tablero con esa modificación en un archivo llamado **02-tablerout.txt**.

Cada pieza tiene un tipo de movimiento asociado, los cuales se muestran en las siguientes imágenes:



Torre



Caballo

Sin embargo, para la **Torre** el movimiento queda limitado a un máximo de **dos cuadros en cada dirección**.

Para el archivo de ejemplo, la interacción por pantalla debería ser:

```
Caballo encontrado en la posición (4, 4).
Los posibles movimientos son:
(2, 3); (2, 5); (6, 3); (6, 5); (3, 2); (5, 2); (3, 6); (5, 6).
Seleccione uno de ellos (fil, col): 6 3
```

Para esta elección el archivo de salida debería ser:

	1	2	3	4	5	6
1	*	*	*	*	*	*
2	*	*	*	*	*	*
3	*	*	*	*	*	*
4	*	*	*	*	*	*
5	*	*	*	*	*	*
6	*	*	C	*	*	*

Salida: 02-tablerout.txt

```
******
******
******
******
******
**C***
```

Utilice funciones o procedimientos para hallar la posición de la pieza en el tablero inicial, así como para calcular los posibles movimientos y el tablero actualizado.