# Tarea 1
## Gato y Ratón
Entrega: martes 12 de abril a las 21:00 hrs.

## 1. Introduccion

El juego del gato y el ratón es un juego de estrategia por turnos, con un poco de suerte. Involucra a dos jugadores por partida, ambos jugadores tendrán una ficha en un tablero de 10x10 donde una ficha representa a un gato y la otra a un ratón. Adicionalmente, existen las murallas. Éstas son de dimensión $1\times2$ y son 2 en total (pueden estar tanto en vertical como horizontal). Después de cada turno (es decir, que el gato y el ratón se hayan movido), las murallas cambian aleatoriamente de lugar en el mapa. Las murallas tienen hoyos que son lo suficientemente grandes para que el ratón se esconda del gato y éste no lo pueda atacar. El juego consiste en que el gato debe intentar atrapar al ratón y el ratón debe esconderse en las murallas. Las reglas del juego son las siguientes:

1. El gato solo puede avanzar dos espacios como máximo (se puede mover 1 o 2 espacios por turno) por turno en dirección horizontal o vertical.
2. El ratón solo se puede mover un espacio, pero este puede hacerlo en cualquier dirección (incluso diagonal).
3. En cada turno, el ratón está obligado a moverse.
4. El gato gana cuando logra cazar al ratón. Para esto el gato y el ratón deben estar en la misma casilla.
5. El ratón gana después de lograr haberse escondido 5 veces en las murallas.

## 2. La Tarea

Su objetivo en esta tarea es hacer un programa en `c` que permita a dos jugadores enfrentarse. Se deberá implementar en 2 modalidades, un modo *versus* —entre dos personas— y un modo *automatizado* que permita enfrentarse a un jugador controlado por el computador. Para lograr esto, se debe implementar lo siguiente:

1. Reciba el modo de juego por la consola[1] y en caso de ser automático, suponga que el computador siempre sera el ratón

[1] -v para versus y -a para el modo automatizado

2. El programa al iniciar el juego debe posicionar al ratón, gato y murrallas de forma aleatoria y que estas no se superpongan y esten en una posicion valida(ejemplo: Si el ratón esta en (5,E) las murrallas o el gato no piueden estar en (5,E))

3. Imprima el tablero actualizado en cada turno. La Figura 1 muestra un tablero en juego. En ella, `[_]` será un espacio vacio, `[G]` será el gato y `[R]` será el ratón y `[M]` serán las murallas.

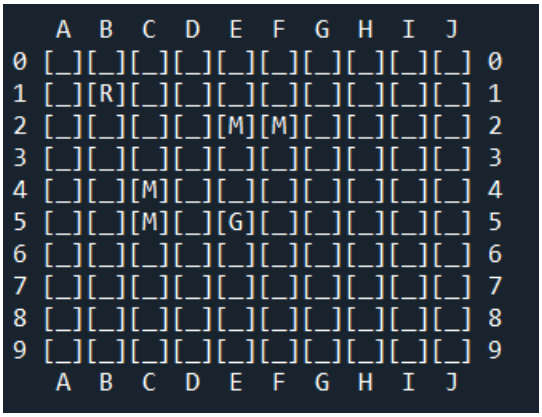


Figura 1: Ejemplo de tablero juego

4. Pida a cada jugador una coordenada para avanzar (p.ej.: `1,A` Siendo 1 el lado de las filas y A de las columnas), en el caso de estar jugando contra la computadora, ambos valores deben ser generados de manera aleatoria[2]. Las coordenadas ingresadas por el jugador deben ser validadas (su programa deberá verificar que el movimiento esté permitido). Las coordenadas dadas por el jugador no pueden ser las mismas en la que está actualmente.

5. En el caso de multijugador, el jugador 1 será el ratón y el jugador 2 será el gato (jugador 1 es quien parte jugando)

6. Además el gato tiene el poder de congelar al ratón por una única vez por partida. Esto quiere decir que el ratón no podrá moverse por un turno. El jugador que controle al gato podrá activarlo ingresando la palabra "FREEZE" después de su jugada (p.ej.: "1,B FREEZE").

7. Repetir los turnos hasta que uno de los dos jugadores gane.

## 3. Evaluación y Entrega

El plazo para la entrega de la tarea vence impostergablemente el martes 12 de abril a las 21:00 hrs.

[2] **hint:** para esto se puede usar la funcion `rand()` de `stdlib.h`

Formato de entrega: Subir un solo archivo con todo el código fuente de su programa al módulo correspondiente a la tarea en la página del curso en Canvas, con el nombre de archivo "APELLIDO1-APELLIDO2-Tarea1.c", reemplazando "APELLIDO1" y "APELLIDO2" según corresponda (ej. PRAT-O'HIGGINS-Tarea1.c). Los archivos compilados no serán tomados en cuenta, si se llega a subir solo un archivo compilado, este será ignorado, y evaluado con nota 1.

Su tarea deberá compilar sin *warnings*, con el estándar más estricto. Para asegurarse de ello, llame al compilador de la siguiente forma:

```
gcc -std=c99 -Wall -Wextra -Wundef -Werror -Wuninitialized -Winit-self archivo.c -o salida
```

Esto obligará al compilador a no saltarse ningún *warning* y a tratarlos como si fueran errores.

Aquellas tareas que no compilen de la forma normal serán evaluadas con nota 1. Las que compilen, pero se caigan durante la ejecución, serán evaluadas con nota máxima 3.

Su programa podría ser evaluado con múltiples casos de prueba y deberá ser capaz de ejecutarlos todos de manera correcta. De fallar en algún caso de prueba serán descontados los puntos correspondientes a dicho caso.

## 4. Consideraciones de Trabajo

- Trabajo en parejas.
- Su programa debe ser robusto frente a datos erróneos. Esto quiere decir que su programa no debe caerse en caso de que el usuario ingrese coordenadas inválidas. Sin embargo, pueden asumir que el usuario será amigable, en el sentido de que cuando le soliciten un número no ingresará letras, por ejemplo.
- El trabajo en esta tarea debe ser hecho solo por Ud. o su grupo. Cuide su tarea para que no sea copiada parcial o íntegramente por otros. Todas las tareas entregadas serán comparadas por un sistema automático de detección de plagio. Cualquier acto contrario a la honestidad intelectual será informado y sometido a proceso conforme al procedimiento disciplinario del reglamento del alumno.