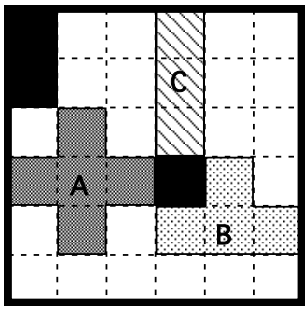
NOMBRE Y APELLIDOS: _____

Ejercicios en Blanco: 1 2 3 4 5 (Rodea con un círculo los ejercicios no entregados)

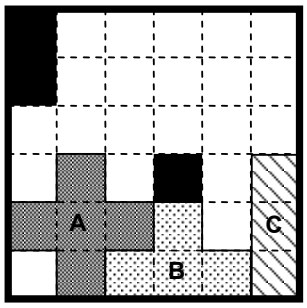
TIEMPO DE REALIZACIÓN: 3 HORAS

Se ha de entregar **cada ejercicio en folios distintos**. Cada folio debe incluir el **nombre del alumno y el número** del ejercicio **y estar numerado cuando hay más de una hoja por ejercicio**.

1. Formalización de un problema de Búsqueda (3,5 puntos)



Estado Inicial



Estado Objetivo

Esta figura 1 muestra el estado inicial de un puzle de dos dimensiones con tres piezas (A, B y C) que pueden moverse en las cuatro direcciones siempre que los límites del tablero y las casillas negras (obstáculos del tablero que no se mueven) no se lo impidan.

El problema debe plantearse como un problema de búsqueda en un espacio de estados para encontrar una solución óptima, si existe, dado cualquier estado inicial posible y que pueda ser resuelto usando la implementación de cualquier estrategia de búsqueda de las vistas en clase. Realiza la implementación en C completa (ficheros .h y .c) para la formalización de este problema:

A. Constantes necesarias para este problema, lista de operadores, tipos de datos necesarios para este problema, en particular el tipo de datos tEstado apropiado para este problema.
B. Funciones para poder crear los estados inicial y final.
C. Función esVálido que determina si es o no posible la aplicación de cada operador a partir de un estado concreto **solamente para 2 operadores de la pieza B**.
D. Función aplicaOperador que lleva a cabo la aplicación de cualquiera de los posibles operadores devolviendo un nuevo estado **solamente para 2 operadores de la pieza B**.
E. Función TestObjetivo para comprobar si se ha alcanzado el objetivo del problema de acuerdo al enunciado propuesto.
F. Función heurística admisible.
G. Construye el árbol de búsqueda correspondiente a la estrategia en Anchura hasta nivel 3

2. Aplicación de Minimax y Poda alfa-beta (1,5 puntos)

Para el juego del tic tac toe visto en clase, y suponiendo el estado inicial dado en la figura 3 donde es el turno de MAX, el cual juega con las fichas X, determina cuál sería la mejor jugada para MAX tras aplicar las dos estrategias, Minimax y Minimax con Poda alfa-beta. Las celdas están numeradas de 0 a 8, empezando de izquierda a derecha, y de arriba abajo, tal y como aparece en la figura 3.

X	O	X
3	O	5
O	7	8

Figura 3

Elige una de las dos opciones aplicando las jugadas de menor a mayor numeración:

- **Construye el árbol de búsqueda completo** suponiendo el estado inicial de la figura 3, y siendo el turno de MAX.

- **Diseña una función de evaluación heurística** y genera el árbol de profundidad 2 (un turno para cada jugador) suponiendo el estado inicial de la figura 3, y siendo el turno de MAX.

3. Implementación en CLIPS de un Sistema Basado en Reglas (2,5 puntos)

Se va a realizar la simulación de un sistema que realiza el control de distintas acciones sobre una aeronave dados diferentes factores. En particular nos centraremos en la autorización por parte de un aeródromo para realizar la maniobra de despegue y para determinar el tiempo estimado de vuelo cuando se alcanza la velocidad de crucero.

Para ello en el aeródromo de origen y de destino se ha de disponer de información relativa a la aeronave, al piloto asignado para su control, al aeródromo y de otras características del vuelo entre dos puntos en general.

3.1. Define las plantillas necesarias para cada una de estas entidades:

- Las **aeronaves** cuentan con diferente información para identificar la aeronave, la compañía, los aeródromos de origen y destino, la velocidad actual así como dos campos que deben tomar una serie de valores específicos. El primer campo con valores enumerados será la Petición que realiza la aeronave para realizar una determinada acción: Ninguna, Despegue, Aterrizaje, Emergencia, Intecerptacion, Informacion, Rumbo. El segundo campo hace referencia al estado actual de la aeronave: *enTierra* (valor por defecto), *Ascenso*, *Crucero*, *Descenso*.
- Los **aeródromos**, que tienen un identificador, la ciudad donde se encuentran, el estado del Radar, que puede tomar los valores ON cuando funciona correctamente y OFF en cualquier otra situación, y varios parámetros relativos a las condiciones atmosféricas, como son el radio de Visibilidad, medido en kms y la velocidad del viento, medida también en km/h.
- El **piloto** asignado a una determinada aeronave para realizar un vuelo determinado, y que confirma la acción requerida por una aeronave a través de un campo, estado, cuyos valores permitidos son: *OK, SOS, Ejecutando, Stand-by* (valor por defecto).
- Información general sobre los **vuelos** entre dos aeródromos (sus identificadores), donde se especifica la distancia, la velocidad de despegue (por defecto 240 km/h) y la velocidad de crucero estándar medida de acuerdo a una serie de parámetros de los vuelos comerciales típicos (no entramos en más detalles en este enunciado), por defecto 700 km/h.

3.2. Reglas: No se admiten instrucciones *if then else* en los antecedentes de las reglas. Todas las reglas deben imprimir en pantalla el resultado de las acciones realizadas lo más detallado posible, por ejemplo, para una acción de despegue:

El vuelo FX001 de la compañía IBERIA va a realizar la acción despegue desde el aeródromo UB34 de Jerez con destino Madrid.

Despegar: se realizará esta acción cuando una aeronave que se encuentra en tierra haya realizado esta petición al aeródromo de origen, el piloto de la misma ha dado su visto bueno (estado *OK*) y en el aeródromo de origen el radar funciona correctamente, el radio de visibilidad es mayor de 5 kms y la velocidad del viento es menor de 75km/h. La autorización de despegue implica que el piloto pasa al estado de *Ejecutando* (esta acción) y la aeronave al estado *Ascenso*. La velocidad actual debe tomar el valor de la velocidad de despegue establecida para este vuelo. Se actualiza la petición de la aeronave a *Ninguna*.

Excepción: Cuando no hay un piloto asociado a una aeronave para realizar un vuelo de los registrados en el aeródromo de origen y la aeronave se encuentra en petición de Despegue. La aeronave realiza una petición de Emergencia mostrando un mensaje que indique este estado de excepción.

Crucero: La velocidad de crucero se alcanza después de que el piloto ha realizado una maniobra de despegue, la aeronave se encuentra en el estado *Ascenso* y pasa a *Crucero*, donde, a partir de la velocidad inicial se alcanza la altura y velocidad de crucero establecidas para este vuelo.
En este momento se informa a los pasajeros de que el despegue ha sido correcto y se estima el tiempo de vuelo, que se calcula con la distancia al destino y la velocidad de crucero alcanzada. (Se ha de crear una función que devuelva esta estimación). El estado del piloto pasará a *stand-by*.
Función tiempo estimado: Crear dos funciones en Clips para calcular el tiempo para llegar al destino, según la velocidad de crucero y la distancia en kms.
Una función ha de devolver el número de horas y la otra los minutos. Por ejemplo una aeronave que va a velocidad de crucero 800 km/h tardará en recorrer 880 kms 1 hora y 6 minutos.
(Puedes usar las funciones div y mod si las necesitas)

4. RETE (1,5 puntos)

Dado la siguiente Base de Reglas y la Base de Hechos, de acuerdo a las plantillas definidas y suponiendo que existe en el sistema una función llamada siguiente_pasillo que devuelve el siguiente pasillo al actual o el primer pasillo cuando se llega al número 11 (esta función no es necesario implementarla ni incluirla en el gráfico):

1. Construye la Red de Redundancia Temporal (RETE).
2. Realiza una simulación de la ejecución correspondiente con la red construida

Plantillas de hechos

```
(deftemplate producto
  (slot id_producto)
  (slot nombre)
  (slot pasillo)
  (slot stock)
  (slot precio)
)
```

```
(deftemplate pedido
  (slot id_cliente)
  (slot id_producto)
  (slot unidades (default 1))
)
```

```
(deftemplate carro
  (slot id_cliente)
  (slot factura (default  0))
  (slot num_productos (default  0))
  (slot pasillo_actual(default  1))
)
```

Base de Reglas

```
(defrule R1_asignar_carro
  (nuevo_cliente ?c1)
  (not (carro (id_cliente ?c1)))
=>
  (assert (carro (id_cliente ?c1) ))
)
```

```
(defrule R2_mover
  (pedido (id_cliente ?c1) (id_producto ?p1))
  ?f1<-(carro (id_cliente ?c1) (pasillo_actual ?pa1))
  (producto (id_producto ?p1) (pasillo ?pa2&~?pa1))
=>
  (modify ?f1 (pasillo_actual (siguientePasillo ?pa1)))
```

	);; mover2

```
(defrule R3_comprar
  ?f1<-(producto (pasillo ?pa) (id_producto ?pr) (stock ?s) (precio ?pu))
  ?f2<-(pedido (id_cliente ?c) (id_producto ?pr) (unidades ?u))
  ?f3<-(carro (pasillo_actual ?pa) (id_cliente ?c) (factura ?f) (num_productos ?np))
  (test (>= ?s ?u)) ;; hay cantidad suficiente
=>
  (modify ?f1 (stock (- ?s ?u)))
  (retract ?f2)
(modify ?f3 (num_productos (+ ?np ?u)) (factura (+ ?f  (* ?pu ?u))) )
)
```

Base de Hechos

```
(producto (id_producto 1) (nombre leche) (pasillo 3)  (stock 25) (precio 0.9))
(pedido (id_cliente 33) (id_producto 1) (unidades 4))
(nuevo_cliente  33)
(nuevo_cliente 11)
(producto (id_producto 5) (nombre galletas) (pasillo 2)  (stock 50) (precio 1.5))
(pedido (id_cliente 11) (id_producto 5) (unidades 1))
```

5. **Implementa en lenguaje C la estrategia de búsqueda informada A* de forma que pueda ser ejecutada con cualquiera de las formalizaciones de problemas de búsqueda vista en clase, en particular, con el 8-puzle.**

(código principal de la búsqueda A* y la función Expandir Nodos, junto con las funciones auxiliares necesarias para su correcta ejecución) (1 punto)