

# Índice

<b>1. Parte 1</b>	<b>2</b>
1.1. Ejercicio 1 . . . . .	2
1.2. Ejercicio 2 . . . . .	3
1.3. Ejercicio 3 . . . . .	3
1.4. Ejercicio 4 . . . . .	3
1.5. Ejercicio 5 . . . . .	4
1.6. Ejercicio 6 . . . . .	4
1.7. Ejercicio 7 . . . . .	4
1.8. Ejercicio 8 . . . . .	4
<b>2. Parte 2</b>	<b>5</b>
2.1. Ejercicio 9 . . . . .	5
2.2. Ejercicio 10 . . . . .	6
2.3. Ejercicio 11 . . . . .	6
2.4. Ejercicio 12 . . . . .	6
2.5. Ejercicio 13 . . . . .	6
2.6. Ejercicio 14 . . . . .	7
2.7. Ejercicio 15 . . . . .	7
2.8. Ejercicio 16 . . . . .	7
2.9. Ejercicio 17 . . . . .	7

# 1. Parte 1

## Predicados y auxiliares generales:

- $\text{aux min } (a, b: \mathbb{Z}) : \mathbb{Z} = \text{if } a < b \text{ then } a \text{ else } b \text{ fi};$
- $\text{aux max } (a, b: \mathbb{Z}) : \mathbb{Z} = \text{if } a > b \text{ then } a \text{ else } b \text{ fi};$
- $\text{aux casillaVacía } (p: \text{posicion}, c: \text{coordenada}) : \text{Bool} = p_0[c_0][c_1] = (0, 0);$
- $\text{aux coordenadaEnRango } (c: \text{coordenada}) : \text{Bool} = 0 \leq c_0, c_1 \leq 7;$
- $\text{aux jugadoresCorrectos } (p: \text{posicion}) : \text{Bool} = 1 \leq p_1 \leq 2;$
- $\text{aux jugadorEn } (p: \text{posicion}, c: \text{coordenada}) : \mathbb{Z} = p_0[c_0][c_1]_1;$
- $\text{aux piezaEn } (p: \text{posicion}, c: \text{coordenada}) : \mathbb{Z} \times \mathbb{Z} = p_0[c_0][c_1];$
- $\text{aux jugadorEnTurno } (p: \text{posicion}) : \mathbb{Z} = p_1;$
- $\text{aux jugadorOponente } (p: \text{posicion}) : \mathbb{Z} = \text{if } p_1 = 1 \text{ then } 2 \text{ else } 1 \text{ fi};$
- $\text{aux reyBlanco} : \mathbb{Z} \times \mathbb{Z} = (4, 1);$
- $\text{aux reyNegro} : \mathbb{Z} \times \mathbb{Z} = (4, 2);$
- $\text{aux peonBlanco} : \mathbb{Z} \times \mathbb{Z} = (1, 1);$
- $\text{aux peonNegro} : \mathbb{Z} \times \mathbb{Z} = (1, 2);$
- $\text{aux torreBlanca} : \mathbb{Z} \times \mathbb{Z} = (3, 1);$
- $\text{aux torreNegra} : \mathbb{Z} \times \mathbb{Z} = (3, 2);$

## 1.1. Ejercicio 1

```
pred esPosicionValida (p: posicion) {  
  tableroConDimensionesCorrectas(p)  
  ∧ jugadoresCorrectos(p)  
  ∧ tableroConCasillasCorrectas(p)  
  ∧ hayUnSoloReyJ(p, 1)  
  ∧ hayUnSoloReyJ(p, 2)  
}
```



## Predicados auxiliares:

- $\text{pred tableroConDimensionesCorrectas } (p: \text{posicion}) \{$   
   $|p_0| = 8 \wedge (\forall i: \mathbb{Z})(0 \leq i < |p_0| \longrightarrow_L |p_0[i]| = 8)$   
   $\}$
- $\text{pred tableroConCasillasCorrectas } (p: \text{posicion}) \{$   
   $(\forall i: \mathbb{Z})(\forall j: \mathbb{Z})((0 \leq i < |p_0| \wedge 0 \leq j < |p_0[i]|) \longrightarrow_L (p_0[i][j] = (0, 0) \vee (1 \leq p_0[i][j]_0 \leq 4 \wedge 1 \leq p_0[i][j]_1 \leq 2)))$   
   $\}$
- $\text{pred hayUnSoloReyJ } (p: \text{posicion}, j: \text{jugador}) \{$   
   $(\exists a: \text{coordenada})(\text{coordenadaEnRango}(a) \wedge \text{piezaEn}(p, a) = (4, j) \wedge$   
   $(\forall b: \text{coordenada})(b \neq a \longrightarrow_L \text{piezaEn}(p, b) \neq (4, j)))$   
   $\}$



## 1.2. Ejercicio 2

```

pred esPosicionInicial (p: posicion) {
  jugadorEnTurno(p) = 1
  ∧ tableroConDimensionesCorrectas(p)
  ∧ p[0] = seq⟨(3, 2), (0, 0), (2, 2), (0, 0), (4, 2), (2, 2), (0, 0), (3, 2)⟩
  ∧ (∀j : ℤ)(0 ≤ j < 8 ⟶L p[1][j] = peonNegro)
  ∧ (∀i : ℤ)(∀j : ℤ)(2 ≤ i < 6 ∧ 0 ≤ j < 8 ⟶L p[i][j] = (0, 0))
  ∧ (∀j : ℤ)(0 ≤ j < 8 ⟶L p[6][j] = peonBlanco)
  ∧ p[7] = seq⟨(3, 1), (0, 0), (2, 1), (0, 0), (4, 1), (2, 1), (0, 0), (3, 1)⟩
}

```

Ese ^ debería ser un ^L porque si el tablero no tiene dimensiones correctas (por ej si es de 4x4) se podrían indefinir las líneas siguientes

## 1.3. Ejercicio 3

```

pred esMovimientoValido (p: posicion, o: coordenada, d: coordenada) {
  esPosicionValida(p)
  ∧ coordenadaEnRango(o) ∧ coordenadaEnRango(d)
  ∧L ¬casillaVacía(p, o) ∧ casillaVacía(p, d)
  ∧ (p[o][o1]0 ≠ 1 ∨ movimientoValidoPeon(p, o, d))
  ∧ (p[o][o1]0 ≠ 2 ∨ movimientoValidoAlfil(p, o, d))
  ∧ (p[o][o1]0 ≠ 3 ∨ movimientoValidoTorre(p, o, d))
  ∧ (p[o][o1]0 ≠ 4 ∨ movimientoValidoRey(o, d))
}

```

## 1.4. Ejercicio 4

```

pred esCapturaValida (p: posicion, o: coordenada, d: coordenada) {
  esPosicionValida(p)
  ∧ coordenadaEnRango(o) ∧ coordenadaEnRango(d)
  ∧L ¬casillaVacía(p, o) ∧ ¬casillaVacía(p, d)
  ∧ jugadorEn(p, o) ≠ jugadorEn(p, d)
  ∧ (p[o][o1]0 ≠ 1 ∨ movimientoValidoPeon(p, o, d))
  ∧ (p[o][o1]0 ≠ 2 ∨ movimientoValidoAlfil(p, o, d))
  ∧ (p[o][o1]0 ≠ 3 ∨ movimientoValidoTorre(p, o, d))
  ∧ (p[o][o1]0 ≠ 4 ∨ movimientoValidoRey(o, d))
}

```

Ahí sería capturaValidaPeon

## Predicados auxiliares para ejercicios 3 y 4:

```

▪ pred movimientoValidoPeon (p: posicion, o: coordenada, d: coordenada) {
  o1 = d1 ∧ ((jugadorEn(p, o) = 1 ∧ d0 = o0 - 1) ∨ (jugadorEn(p, o) = 2 ∧ d0 = o0 + 1))
}

```

```

▪ pred capturaValidaPeon (p: posicion, o: coordenada, d: coordenada) {
  (jugadorEn(p, o) = 1 ∧ d0 = o0 - 1 ∧ abs(o1 - d1) = 1)
  ∨ (jugadorEn(p, o) = 2 ∧ d0 = o0 + 1 ∧ abs(o1 - d1) = 1)
}

```

```

▪ pred movimientoValidoAlfil (p: posicion, o: coordenada, d: coordenada) {
  abs(o0 - d0) = abs(o1 - d1) ∧
  (∀x : ℤ)(min(o0, d0) < x < max(o0, d0) ⟶L
  (∀y : ℤ)(min(o1, d1) < y < max(o1, d1) ⟶L
  ¬(abs(o0 - x) = abs(o1 - y) ∧ abs(d0 - x) = abs(d1 - y)) ∨
  p[x][y] = (0, 0))
}

```

```

▪ pred movimientoValidoTorre (p: posicion, o: coordenada , d: coordenada) {
  (o0 = d0 ∧ (∀y : Z)(min(o1, d1) < y < max(o1, d1) →L casillaVacía(p, (o0, y)))) ∨
  (o1 = d1 ∧ (∀x : Z)(min(o0, d0) < x < max(o0, d0) →L casillaVacía(p, (x, o1))))
}

▪ pred movimientoValidoRey (o: coordenada , d: coordenada) {
  abs(o0 - d0) ≤ 1 ∧ abs(o1 - d1) ≤ 1 ∧ abs(o0 - d0) + abs(o1 - d1) > 0
}

```

## 1.5. Ejercicio 5

```

proc casillasAtacadas (in p: posicion, in j: jugador, out atacadas: coordenada) {
  Pre {esPosicionValida(p) ∧ jugadoresCorrectos(p)}
  Post {(∀e : coordenada)(e ∈ atacadas →L ((∃o : coordenada)
    (jugadorEn(p, o) = j ∧ (esMovimientoValido(p, o, e) ∨ esCapturaValida(p, o, e)))
    ∧ cantidadDeApariciones(atacadas, e) = 1))}
}

```

**Predicados auxiliares:**

```

pred ∈ (e: Z × Z, s: seq(Z × Z)) {
  (∃i : Z)(0 ≤ i < |s| ∧ s[i] = e)
}

```

```

aux cantidadDeApariciones (s: seq(Z × Z), e: Z × Z) : Z = ∑i=0|s| (if s[i] = e then 1 else 0 fi);

```

## 1.6. Ejercicio 6

```

proc dondeEstaElRey (in p: posicion, in j: jugador, out c: coordenada) {
  Pre {esPosicionValida(p) ∧ jugadoresCorrectos(p)}
  Post {coordenadaEnRango(c) ∧L piezaEn(p, c) = (4, j)}
}

```

## 1.7. Ejercicio 7

```

proc esPosicionSiguiente (in p1: posicion, in p2: posicion, in o: coordenada, in d: coordenada, out res: Bool) {
  Pre {esPosicionValida(p1) ∧ esPosicionValida(p2) ∧ (esMovimientoValido(p1, o, d) ∨ esCapturaValida(p1, o, d))}
  Post {res = true ↔
    (∀c : coordenada)(coordenadaEnRango(c) ∧ c ≠ o ∧ c ≠ d →L piezaEn(p1, c) = piezaEn(p2, c))
    ∧ esCasillaVacía(p2, o)
    ∧ (piezaEn(p2, d) = piezaEn(p1, o) ∨ (piezaEn(p1, o) = peonBlanco ∧ o0 = 1 ∧ abs(d1 - o1) = 1 ∧ d0 = 0 ∧ piezaEn(p2, d) =
    torreBlanca) ∨
    (piezaEn(p1, o) = peonNegro ∧ o0 = 6 ∧ abs(d1 - o1) = 1 ∧ d0 = 7 ∧ piezaEn(p2, d) = torreNegra))}
}

```

## 1.8. Ejercicio 8

```

proc estaOrdenado (in p: posicion, out res: Bool) {
  Pre {esPosicionValida(p)}
  Post {res = true ↔ (∀i : Z)(0 ≤ i ≤ 7 →L esFilaOrdenada(p0[i]))}
}

```

Tratar de evitar valores "harcodados"

**Predicado auxiliar:**

```

pred esFilaOrdenada (l: seq(casilla)) {
  (∀i : Z)((0 ≤ i < |l| ∧L l[i] ≠ (0, 0)) →L (∀j : Z)(0 ≤ j < i →L l[j]0 ≤ l[i]0))
}

```

## 2. Parte 2

### Predicados y auxiliares generales:


- **pred esPosicionValidaContemplandoJaque** (p: posicion) {  
 $esPosicionValida(p)$   
 $\wedge \neg hayJaqueMate(p)$   
 $\wedge \neg((esJaque(p, 1) \wedge jugadorEnTurno(p) = 2) \vee (esJaque(p, 2) \wedge jugadorEnTurno(p) = 1))$   
 }
- **pred esCoordenadaDelReyJ** (p: posicion, c: coordenada, j: jugador) {  
 $coordenadaEnRango(c) \wedge_L piezaEn(p, c) = (4, j)$   
 }
- **pred esCasillaBajoPotencialCaptura** (p: posicion, c: coordenada) {  
 $(\exists a : coordenada)(jugadorEn(p, a) \neq jugadorEn(p, c) \wedge esCapturaValida(p, a, c))$   
 }
- **pred esJaque** (p : posicion, j: jugador) {  
 $(j = 1 \wedge (\exists c : coordenada)(esCoordenadaDelReyJ(p, c, 1) \wedge_L esCasillaBajoPotencialCaptura(p, c)))$   
 $\vee$   
 $(j = 2 \wedge (\exists c : coordenada)(esCoordenadaDelReyJ(p, c, 2) \wedge_L esCasillaBajoPotencialCaptura(p, c)))$   
 }
- **pred hayJaqueMate** (p: posicion) {  
 $esPosicionValida(p)$   
 $\wedge esJaque(p, p_1)$   
 $\wedge \neg(\exists o, d : coordenada)(coordenadaEnRango(o) \wedge coordenadaEnRango(d) \wedge$   
 $(esMovimientoValido(p, o, d) \vee esCapturaValida(p, o, d)) \wedge jugadorEn(p, o) = jugadorEnTurno(p) \wedge$   
 $(\exists q : posicion)(esPosicionSiguienteConMovimiento(p, q, o, d) \wedge \neg esJaque(p, jugadorEnTurno(p))))$   
 }
- **pred esPosicionSiguienteConMovimiento** (p1: posicion, p2:posicion, o:coordenada, d:coordenada) {  
 $(esMovimientoValido(p1, o, d) \vee esCapturaValida(p1, o, d))$   
 $\wedge (\forall c : coordenada)(coordenadaEnRango(c) \wedge c \neq o \wedge c \neq d \longrightarrow_L piezaEn(p1, c) = piezaEn(p2, c)$   
 $\wedge esCasillaVacía(p2, o)$   
 $\wedge (piezaEn(p2, d) = piezaEn(p1, o) \vee (piezaEn(p1, o) = peonBlanco \wedge o_0 = 1 \wedge abs(d_1 - o_1) = 1 \wedge d_0 = 0 \wedge$   
 $piezaEn(p2, d) = torreBlanca) \vee$   
 $(piezaEn(p1, o) = peonNegro \wedge o_0 = 6 \wedge abs(d_1 - o_1) = 1 \wedge d_0 = 7 \wedge piezaEn(p2, d) = torreNegra))$   
 }

### 2.1. Ejercicio 9

```

proc ordenarTablero (inout p: posicion) {
  Pre {esPosicionValida(p) ∧ p = p̃}
  Post {esTableroOrdenado(p) ∧ jugadorEnTurno(p) = jugadorEnTurno(p̃)}
  ∧ (∀i : Z)(∀j : Z)(1 ≤ i ≤ 4 ∧ 1 ≤ j ≤ 2 ∧ (∃c : coordenada)
    (coordenadaEnRango(c) ∧L piezaEn(p̃, c) = (i, j)) →L (∃k : Z)(0 ≤ k ≤ 7 ∧ p0[c0][k] = (i, j)))
  ∧ (∀d : coordenada)(coordenadaEnRango(d) ∧L casillaVacía(p̃, d) →L casillaVacía(p, d))}
}

```



### Predicados auxiliares:

```

pred esTableroOrdenado (p: posicion) {
  esPosicionValida(p) ∧ (∀i : Z)(0 ≤ i ≤ 7 →L esFilaOrdenada(p0[i]))
}
pred esFilaOrdenada (l: seq<casilla>) {
  (∀i : Z)((0 ≤ i < |l| ∧L l[i] ≠ (0,0)) →L (∀j : Z)(0 ≤ j < i →L l[j]0 ≤ l[i]0))
}

```

Predicado duplicado

## 2.2. Ejercicio 10

```

proc esJaqueMate (in p: posicion, out res: Bool) {
  Pre {esPosicionValida(p)}
  Post {esJaque(p, p1) ∧ ¬(∃o, d : coordenada)(coordenadaEnRango(o) ∧ coordenadaEnRango(d) ∧
    (esMovimientoValido(p, o, d) ∨ esCapturaValida(p, o, d)) ∧ jugadorEn(p, o) = jugadorEnTurno(p) ∧
    (∃q : posicion)(esPosicionSiguienteConMovimiento(p, q, o, d) ∧ ¬esJaque(p, jugadorEnTurno(p))))}
}

```



## 2.3. Ejercicio 11

```

proc esEmpate (in p: posicion, out res: Bool) {
  Pre {esPosicionValida(p)}
  Post {esMateAhogado(p) ∨ ambosReyesSolos(p)}
}

```



Cuando un proc devuelve un Bool deberían en la Post tener algo de la forma: res = true <-> condición.

Si no, res no está siendo condicionada por nada. Es decir puede cumplirse todo lo necesario para que sea empate, por ejemplo, pero devolver res = false y va a cumplirse la Post. A su vez, si no es empate no va a valer la Post y esto no es lo que se quiere, se quiere que valga la Post pero que res sea false.

Esto aplica para varios ejercicios.

### Predicados auxiliares:

```

pred esMateAhogado (p : posicion) {
  (∃j : jugador)((jugadorEnTurno(p) = j) ∧ jugadoresCorrectos(p) ∧ (¬esJaque(p, j)) ∧
    (∀o, d : coordenada)(coordenadaEnRango(o) ∧ coordenadaEnRango(d) ∧ jugadorEn(p, o) = jugadorEnTurno(p) →L
    ¬esMovimientoValido(p, o, d) ∧ ¬esCapturaValida(p, o, d)))
}
pred ambosReyesSolos (p : posicion) {
  (∀c : coordenada)(coordenadaEnRango(c) →L p0[c0][c1]0 = 0 ∨ p0[c0][c1]0 = 4)
}

```

esPosicionValida ya verifica jugadoresCorrectos(p)



## 2.4. Ejercicio 12

```

pred esJugadaLegal (p : posicion, o: coordenada, d: coordenada) {
  esPosicionValidaContemplandoJaque(p) ∧ coordenadaEnRango(o) ∧ coordenadaEnRango(d)
  ∧ jugadorEn(p, o) = jugadorEnTurno(p) ∧ jugadorEn(p, d) ≠ jugadorEnTurno(p)
  ∧ (∀q : posicion)(esPosicionSiguienteConMovimiento(p, q, o, d) →L ¬esJaque(q, p1))
}

```



Esto debería ser parte de la Post, puede venir una secuencia de coordenadas cualquiera. Si entre esas coordenadas hay algunas que no tienen sentido por ejemplo por tener piezas de otro jugador se devolverá false en res

## 2.5. Ejercicio 13

```

proc piezasMovibles (in p: posicion, in movibles: seq<coordenada>, out res: Bool) {
  Pre {esPosicionValidaContemplandoJaque(p) ∧
    (∀c : coordenada)(c ∈ movibles →L coordenadaEnRango(c) ∧L jugadorEn(p, c) = jugadorEnTurno(p))}
  Post {res = true ↔ (∀c : coordenada)(coordenadaEnRango(c) →L
    (c ∈ movibles) ↔ (∃d : coordenada)(coordenadaEnRango(d) ∧ esJugadaLegal(p, c, d)))}
}

```

El chequeo de la Post está muy bien

## 2.6. Ejercicio 14

```

proc esPosicionFutura (in p1: posicion, in p2: posicion, out res: Bool) {
  Pre {esPosicionValidaContemplandoJaque(p1) ∧ esPosicionValidaContemplandoJaque(p2)}
  Post {res = true ↔ (∃s : seq<posicion>))
    (s[0] = p1 ∧ s[|s| - 1] = p2 ∧ (∀i : Z)(0 ≤ i < |s| - 1 →L (∃o, d : coordenada)
    (esPosicionSiguienteLegal(s[i], s[i + 1], o, d))))}
}

```

**Predicados auxiliares:**

```

pred esPosicionSiguienteLegal (p: posicion, q: posicion) {
  esPosicionSiguienteConMovimiento(p, q, o, d) ∧ esJugadaLegal(p, o, d)
}

```

## 2.7. Ejercicio 15

```

proc hayJaqueDescubierto (in p: posicion, out res: Bool) {
  Pre {esPosicionValidaContemplandoJaque(p)}
  Post {res = true ↔ (∃o, d : coordenada)(coordenadaEnRango(o) ∧ coordenadaEnRango(d)
    ∧ jugadorEn(p, o) = jugadorEnTurno(p)
    ∧ esJugadaLegal(p, o, d)
    ∧ (∃q : posicion)(∃b, c : coordenada)(esPosicionSiguienteConMovimiento(p, q, o, d) ∧ esJaque(q, jugadorOponente(p))
    ∧ coordenadaEnRango(b) ∧ coordenadaEnRango(c) ∧ b ≠ d ∧ jugadorEn(q, b) = jugadorEnTurno(p) ∧
    esCoordenadaDelReyJ(p, c, jugadorOponente(p)) ∧ esCapturaValida(q, b, c)))}
}

```

Las condiciones siguientes no fuerzan a esJaque? Creo que no es necesario

## 2.8. Ejercicio 16

Rehacer

```

proc hayMateEn1 (in p: posicion, out res: Bool) {
  Pre {esPosicionValidaContemplandoJaque(p)}
  Post {res = true ↔
    (∀q : posicion)(∀o, d : coordenada)(coordenadaEnRango(o) ∧ coordenadaEnRango(d) ∧ esPosicionSiguiente(p, q, o, d)
    →L hayJaqueMate(q))}
}

```

Acá están pidiendo que al mover el jugador actual cualquiera de sus piezas de forma legal se genera un jaque. Pero lo que pedía el ejercicio es que el jaque aparezca en la jugada siguiente: juega el jugador actual, juega el contrincante y pone en jaque al jugador actual.

## 2.9. Ejercicio 17

Rehacer

```

proc ejecutarSecuenciaForzada (inout p: posicion, in s: seq<coordenada × coordenada>) {
  Pre {p = p̃ ∧ esPosicionValidaContemplandoJaque(p) ∧ jugadorEn(p, s[0]0) = jugadorEnTurno(p)
    ∧ (∀j : Z)(0 ≤ j < |s| →L coordenadaEnRango(s[j]0) ∧ coordenadaEnRango(s[j]1))}
  Post {(∃pos : seq<posicion>)(|pos| = 2 * |s| + 1 ∧ pos[0] = p̃ ∧ pos[|pos| - 1] = p
    ∧ (∀i : Z)(0 ≤ i < |pos| →L
    (i mod 2 = 1 ∧ esPosicionSiguienteLegal(pos[i - 1], pos[i], s[i]0, s[i]1)) ∨
    (i mod 2 = 0 ∧ i > 0 ∧
    (∃o, d : coordenada)(esPosicionSiguienteLegal(pos[i - 1], pos[i], o, d))
    ∧ jugadoresAlternados(p̃, pos)
    ))}
}

```

i puede llegar hasta 2\*|s| por lo que indexar a s con i se va a indefinir eventualmente. Sería tomar el piso de dividir a i por 2 o que i esté en el rango de s y multiplicar por 2, eso sería más prolijo.

Estos o y d deberían ser únicos porque para cada movida de s el contrincante tiene una única movida legal para hacer.

**Predicados auxiliares:**

```

pred jugadoresAlternados (p: posicion, pos: seq<posicion>) {
  (∀i : Z)(0 ≤ i < |pos| →L
  (i mod 2 = 0 ∧ pos[i]1 = jugadorEnTurno(p)) ∨ (i mod 2 = 1 ∧ pos[i]1 = jugadorOponente(p)))
}

```

Tendrían que chequear en la Pre que s es una secuencia forzada válida, además de tener coordenadas en rango tiene que existir para cada movimiento uno del contrincante único legal que lleve al movimiento siguiente, básicamente lo que chequearon en la Post debería pasar a la Pre. La Post debería solo condicionar lo que se devuelve, en este caso p.