En Gobstones se desea representar una mesa de apuestas de un casino, en donde distintos jugadores apuestan una cantidad de dinero a números que van entre el 1 y el 46.

El juego se divide en etapas, en primer lugar, todos los **jugadores** colocan en la mesa sus apuestas de dinero (una persona puede realizar varias apuestas a distintos números, con diferentes montos de dinero para cada uno, y distintas personas pueden apostar al mismo número). Las apuestas se colocan en la mesa (que será representado por el tablero), sin ningún orden en particular. El tamaño de la mesa es indistinto y desconocido.

Cada jugador distinto va a ser representado mediante un número (id de jugador). Los números no necesariamente son consecutivos y pueden ser cualquiera. El número que representa al jugador es único en el tablero, además, un jugador puede tener en la mesa distintas apuestas (color, número/s, paridad)

En segundo lugar, el **croupier** (siempre hay uno en el tablero) sortea un número, también entre 1 y 46. Al hacerlo, se paga a los ganadores (si hubieran) según el monto apostado: Número paga 6 por cada peso, color o paridad paga 2 por cada peso, apuesta dividida paga 3 por cada peso. El croupier retira el dinero apostado de todos aquellos que no ganaron. Es decir, el croupier recibe dinero de los perdedores, pero también otorga dinero a los ganadores.

Si el croupier no tiene suficiente dinero para efectuar el pago a los ganadores el programa no debe pinchar. Los ganadores reciben toda la plata ganada, aunque el croupier no la tenga (podríamos asumir que recibe un préstamo), pero el croupier quedará en banca rota (sin fondos).

```
type Jugador is record
// PROP: Modela un jugador.
// INV. DE REP: La edad del jugador debe ser mayor a
//  18 años
    field   idDeJugador     //Número
    field   tipoDeApuesta   //TipoDeApuesta
    field   estáApostando   //Booleano
    field   edad            //Número
```

```
type Crupier is record
// PROP: Modela al crupier.
// INV. DE REP:
//  * El número sorteado debe ser un número entre 1 y 46.
//  * El color del número debe ser Rojo o Negro.
    field   númeroSorteado  //Número
    field   colorSorteado   //Color
    field   fondos          //Número
```

```
type ApuestaANúmero is record
//  PROP: Modela una apuesta a un número.
//  INV. DE REP: El número apostado debe ser un
//   número entre 1 y 46
    field   númeroApostado  //Número
    field   montoApostado   //Número
```

```
type ApuestaDividida is record
// PROP: Modela una apuesta a dos números.
// INV. DE REP: Los números apostados deben ser
//  números entre 1 y 46.
    field   primerNúmeroApostado    //Número
    field   segundoNúmeroApostado   //Número
    field   montoApostado           //Número
```

```
type ApuestaAColor is record
// PROP: Modela una apuesta a un color.
// INV. DE REP: El color debe ser Rojo o Negro.
    field   colorApostado   //Color
    field   montoApostado   //Número
```

```
type ApuestaAParOImpar is record
// PROP: Modela una apuesta a número par o impar.
    field   esApuestaAPar   //Booleano
    field   montoApostado   //Número
```

```
function hayJugadorAcá()
// PROP: Indica si hay un jugador en la celda actual.
// PREC: Ninguna.
```

```
function hayCrupierAcá()
// PROP: Indica si hay un crupier en la celda actual.
// PREC: Ninguna.
```

```
type ApuestaAColor is record
// PROP: Modela una apuesta a un color.
// INV. DE REP: El color debe ser Rojo o Negro.
    field   colorApostado   //Color
    field   montoApostado   //Número
```

```
type ApuestaAParOImpar is record
// PROP: Modela una apuesta a número par o impar.
    field   esApuestaAPar    //Booleano
    field   montoApostado    //Número
```

```
function hayJugadorAcá()
// PROP: Indica si hay un jugador en la celda actual.
// PREC: Ninguna.
// TIPO: Booleano
```

```
function hayCrupierAcá()
// PROP: Indica si hay un crupier en la celda actual.
// PREC: Ninguna.
// TIPO: Booleano
```

```
function jugadorAcá()
// PROP: Describe al jugador en la celda actual.
// PREC: Debe haber un jugador en la celda actual.
// TIPO: Jugador
```

```
function crupierAcá()
// PROP: Describe al crupier en la celda actual.
// PREC: Debe haber un crupier en la celda actual.
// TIPO: Crupier
```

```
function apuestaAcá()
// PROP: Describe la apuesta en la celda actual.
// PREC: Debe haber un jugador apostando en la celda
//  actual.
// TIPO: Sobrecargado (ApuestaANumero,
// ApuestaDividida, ApuestaAColor, ApuestaAParOImpar)
```

```
type TipoDeApuesta is variant
// PROP: Modela los tipos de las apuestas
    case ANúmero     {}
    case Dividida    {}
    case AColor      {}
    case AParOImpar  {}
    case SinApuesta  {}
```

Ejercicio 1)

Escriba la función **cantidadDePerdedoresConDividida** que describe cuántos jugadores perdieron con apuesta dividida, teniendo en cuenta que el crupier ya sorteó el número ganador. Para ello se cuenta con la primitiva ***IrACrupier.***

Ejercicio 2)

Escriba la función **haySospechaPorParidad** que indica si el crupier está rodeado en sus direcciones ortogonales por jugadores que apostaron a la paridad como él. Se puede asumir que el cabezal está sobre el crupier. La función debe ser total.

Ejercicio 3)

Escriba el procedimiento **QuitarAGanadorAcá** que quita el dinero apostado al perdedor de la celda actual asumiendo que hay uno. Se pueden considerar las siguientes primitivas para la resolución del ejercicio:

```
procedure SacarJugador()
// PROP: Saca al jugador de la celda actual.
// PREC: Debe haber un jugador en la celda actual.
```

```
procedure PonerJugador_(unJugador)
// PROP: Pone al jugador **unJugador** en la celda actual.
// PREC: Ninguna
// PAR: unJugador: Jugador - El jugador a poner
```

Ejercicio 4)

Escriba el procedimiento **QuitarMayorApuestaDePerdedorDe**, que dado un tipo de apuesta le quita al