```
//Ejercicio 1

function hayActitudSospechosa(){
   direccionActual := minDir()
   sorteoCroupier := numeroGanador()
   while(direccionActual /= maxDir() && puedeMover(direccionActual)){
      Mover(direccionActual)
      hayActitudSospechosaAca := actitudSospechosaAca(sorteoCroupier)
      Mover(opuesto(direccionActual))
      direccionActual := siguiente(direccionActual)
   }
   return(actitudSospechosaAca(sorteoCroupier))
}

function hayActitudSospechosa(){

}

function actitudSospechosaAca(sorteoCroupier){
   return(esSospechoso(sorteoCroupier))
}

function esSospechoso(sorteoCroupier){
   return(sorteoCroupier == numeroAca())
}

//Ejercicio 2

function cantidadDeDineroDePerdedores(){

   IrAPrimeraCeldaEnUnRecorridoAl_Y_(Este, Norte)
   cantidadTotalDeDineroDePerdedores := cantidadDeDineroSiEsPerdedor()
   while(haySiguienteCeldaEnUnRecorridoAl_Y_(Este, Norte)){
      IrASiguienteCeldaEnUnRecorridoAl_Y_(Este, Norte)
      cantidadTotalDeDineroDePerdedores := cantidadTotalDeDineroDePerdedores +
cantidadDeDineroSiEsPerdedor
   }
   return(cantidadTotalDeDineroDePerdedores)
}

function cantidadDeDineroSiEsPerdedor(){
   return(
      choose
         dineroAca() when esPerdedor()
```

```
        0 otherwise
    )
}

function esPerdedor(){
    return(hayJugadorAca() && elJugadorPerdio())
    }

function elJugadorPerdio(){
    return(numeroAca() /= numeroGanador())
}

function numeroGanador(){
    IrACroupier()
    return(numeroAca())
}

procedure IrACroupier() {
    IrAPrimeraCeldaEnUnRecorridoAl_Y_(Este,Norte)
    while(not hayCroupierAcá()) {
        IrASiguienteCeldaEnUnRecorridoAl_Y_(Este,Norte)
    }
}

//Ejercicio 3

procedure Otorgar_DineroAlCrupier(cantidadDeDineroEntregado){
    Sacar_DeDineroAca(cantidadDeDineroEntregado)
    IrACroupier()
    DarDineroAlCroupier(cantidadDeDineroEntregado)
}

procedure SacarDineroAca(cantidadDeDineroASacar) {
    Sacar(dineroAca())
}

//Ejercicio 4

procedure TomarDineroDePerdedores(){
    IrAPrimeraCeldaEnUnRecorridoAl_Y_(Este, Norte)
    dineroSacadoALosPerdedores := cantidadDeDineroSiEsPerdedor()
    while(haySiguienteCeldaEnUnRecorridoAl_Y_(Este, Norte)){
        IrASiguienteCeldaEnUnRecorridoAl_Y_(Este, Norte)
```

```
        dineroSacadoALosPerdedores := dineroSacadoALosPerdedores +
cantidadDeDineroSiEsPerdedor()
    }
    DarElDineroAlCroupier(dineroSacadoALosPerdedores)
}

procedure DarElDineroAlCroupier(dineroAEntregar){
    IrACroupier()
    DarDinero(dineroAEntregar)
}

procedure DarDinero(dineroADar){
    repeat(dineroADar){
        Poner(colorDinero())
    }
}

////////////////// VIAJE INTERGALÁCTICO
//Ejercicio 1

function hayUnPlanetaA_Hacia_(distancia, direccion){
    Mover_VecesAl_(distancia, direccion)
    return(hayUnPlanetaAca())
}

function hayUnPlanetaAca(){
    return(hayBolitas(planeta))
}

function combustibleRestante(){
    /*Propósito. Indica la cantidad de combustible que le queda a la nave.
    Precondición. El cabezal está sobre la nave.
    Tipo. Número.*/
    return(cantidadDeCombustibleRestante())
}

function cantidadDeCombustibleRestante(){
    return(nroBolitas(combustible))
}

//Ejercicio 2

function sePuedeAterrizarA_HaciaEl_(distanciaAPlaneta, direccionAPlaneta){
```

```
    return(hayUnPlanetaA_Hacia_(distanciaAPlaneta, direccionAPlaneta) &&
hayCombustibleRestantePara_AñosLuz(distanciaAPlaneta))
}

function hayCombustibleRestantePara_AñosLuz(cantidadAñosLuz) {

    return (combustibleRestante() >= cantidadAñosLuz)
}

//Ejercicio 3

function hayPlanetaRecto(){
    direccionActual := minDir()
    while(direccionActual /= maxDir() && not hayPlanetaRectoHaciaEl_(dirActual)){

        Mover(direccionActual)


    }
    return(hayPlanetaRectoHaciaEl_(dirActual))//Devuelve porque cortó el While
}

function hayPlanetaRectoHaciaEl_(direccion) {
    Mover(direccion)
    while(puedeMover(direccion) && not hayUnPlanetaAca()){
        Mover(direccion)
    }

    return (hayUnPlanetaAca())
}

//Ejercicio 4

function puedeLaNave_OrbitarAlgunoPorEmergencia(idNave){
    IrALaNaveConId_(idNave)
    return(estaLaNaveEnEmergencia() && esPosibleParaEstaNaveOrbitarAlgunPlaneta())
}

function estaLaNaveEnEmergencia(){
    return(combustibleRestante == 1)
}

function esPosibleParaEstaNaveOrbitarAlgunPlaneta(){
    dirActual := minDir()
```

```
    while(dirActual /= maxDir() && not esPosibleParaEstaNaveOrbitarAl_(dirActual)){
        dirActual := siguiente(dirActual)
    }

    return(esPosibleParaEstaNaveOrbitarAl_(dirActual))
}

function esPosibleParaEstaNaveOrbitarAl_(direccion){
    return(puedeMover(direccion) && esPosibleOrbitarAl_(direccion))
}

function esPosibleOrbitarAl_(direccion){
    Mover(direccion)
    return(hayUnPlanetaRectoAUnAñoLuz())
}

//Ejercicio 5

procedure AteerizarNave_EnPlaneta_SiEsPosible(idNave, idPlaneta) {

    if(puedeLaNave_AterrizarEnElPlaneta_(idNave, idPlaneta)){
        AterrizarNave_EnPlaneta_(idNave, idPlaneta)
    }
}



function puedeLaNave_AterrizarEnElPlaneta_(idNave, idPlaneta) {
    return (distanciaEntreNave_YPlaneta_(idNave, idPlaneta)
            <= combustibleRestanteEnNave_(idNave))
}

function distanciaEntreNave_YPlaneta_(idNave, idPlaneta){
    return(
        distanciaEnXDe_A_(idNave, idPlaneta) + distanciaEnYDe_A_(idNave, idPlaneta)
    )
}

function  distanciaEnXDe_A_(idNave, idPlaneta){
    return(diferenciaEntre_Y_(coordenadaXDeNave_(nave), coordenadaXDePlaneta_(planeta)))
}

function diferenciaEntre_Y_(primeraCoord, segundaCoord){
    return(
```

```
        choose
            primeraCoord - segundaCoord when (primeraCoord >= segundaCoord)
            segundaCoord - primeraCoord otherwise
        }
    )
}

function coordenadaXDeNave_(idNave){
    IrALaNaveConId_(idNave)
    return(coordenadaX())
}

function combustibleRestanteEnNave_(idNave){

}

procedure AterrizarNave_EnPlaneta_(idNave, idPlaneta) {
    IrALaNaveConId_(idNave)
    SacarLaNave(idNave)
    IrAPlanetaConId_(idPlaneta)
    AterrizarNave_(idNave)

}

///////SHARIKI 5
//Ejercicio 1:

function nivelDeDificultad() {
    IrAPrimeraCeldaEnUnRecorridoAl_Y_(Este, Norte)
    cantidadDeObstaculosEnElTablero := unoSi_CeroSiNo(hayObstaculosAca())
    while(haySiguienteCeldaEnUnRecorridoAl_Y_(Este, Norte))
{
        IrASiguienteCeldaEnUnRecorridoAl_Y_(Este, Norte)
        cantidadDeObstaculosEnElTablero := cantidadDeObstaculosEnElTablero +
unoSi_CeroSiNo(hayObstaculosAca())

    }
    return (cantidadDeObstaculosEnElTablero)
}
function hayObstaculosAca(){
    return (hayBolitas(Negro) && not hayBolitas(Azul))
}

//Ejercicio 2:
```

```
function cantidadDeFichasCirculoEnEstaFila(){
   IrAlBorde(Oeste)
   cantidadDeFichasCirculo := unoSi_CeroSiNo(hayCiruloAca())
   while(puedeMover(Este)){
      Mover(Este)
      cantidadDeFichasCirculo := cantidadDeFichasCirculo + unoSi_CeroSiNo(hayCiruloAca())

   }
   return (cantidadDeFichasCirculo)
}
function hayCiruloAca(){
   return(nroBolitas(Negro)==1 && nroBolitas(Azul)==2)
}

//Ejercicio 3

function hayFichasExplotadas(){
   IrAPrimeraCeldaEnUnRecorridoAl_Y_(Este, Norte)
   while(haySiguienteCeldaEnUnRecorridoAl_Y_(Este, Norte) && not hayFichaExplotada){
      IrASiguienteCeldaEnUnRecorridoAl_Y_(Este, Norte)

   }
   return (hayFichaExplotada())
}

function hayFichaExplotada(){
   return(esVacía())
}
```