CORRECCION PARCIAL:

1) Escribir la función hayTripulanteAcá que indica si en la zona actual hay un tripulante (recordar que los impostores son jugadores, pero no tripulantes…).

Esta pregunta debe contestarse en una hoja de papel (una única carilla), tomarle una foto que esté en foco, y subirla en el lugar de la entrega.

```
Nombre: Pablo Corvera
Ejercicio 1
function hayTripulanteAca(){
/*PROPOSITO:indica si en la zona actual hay un tripulante
PRECONDICION:no tiene
RESULTADO:bool
*/
   return(hayJugador() && not hayImpostor())

}



function hayJugador(){
   return(not hayBolitas(colorJugador())
}

function hayImpostor(){
   return(not hayBolitas(colorImpostor()))
}
```

Comentarios generales:

Como comentarios generales decir que en varias ocaciones haces cosas distintas a las pedidas en el enunciado. Por otro lado, hay graves errores en algunos lados, donde utilizas expresiones como comandos, o donde invocas procedimientos que toman parámetros sin argumentos.

Comentarios del ejercicio:

Debías entregar algo escrito de tu puño y letra, no escrito en la máquina.

Las funciones auxiliares deberían tener contrato también.

El nombre del tipo es "Booleano", no "bool".

Para que efectivamente "haya un impostor" deben haber bolitas. Vos indicas que "no deben haber bolitas". Lo mismo vale para "hayImpostor".

Escribir la función cantidadDeTripulantesEnLaNave que describe la cantidad total de tripulantes que hay en la nave.

2) Escribir la función cantidadDeTripulantesEnLaNave que describe la cantidad total de tripulantes que hay en la nave.

Codigo:

```
function cantidadDeTripulantesEnLaNave(){
/*PROPOSITO:Describe la cantidad de tripulantes totales
que hay en la nave
PRECONDICION:No hay
RESULTADO:Numero
OBSERVACION:se considera que la funcion hayTripulanteAca()
(ejercicio1)
esta ya implementada
*/
    cantidadDeTripulantesVistos:=0
    IrAlInicioDeLaNave__(Norte,Este)
    while (not estoyAlFinalDeLaNave(Norte,Este)){
        cantidadDeTripulantesVistos:=cantidadDeTripulantesVistos
                                    + contarSiHayTripulante()
        DarUnPasoHacia__(Norte,Este)
    }
```

```
    return(cantidadDeTripulantesVistos +
contarSiHayTripulante())
}

function contarSiHayTripulante(){
/*PROPOSITO:cuenta si hay un tripulante en la zona
PRECONDICION:No hay
RESULTADO:Numero*/

    return(choose 1 when hayTripulanteAca()
                  0 otherwise)
}

procedure IrAlIncioDeLaNave__(dirPrincipal, dirSecundaria){
    /*  PROPÓSITO:
            #Posicionar al cabezal en inicio de la nave
        PARÁMETROS:
            #**dirPrincipal**  = Dirección - Indica la dirección
principal de la nave
            #**dirSecundaria** = Dirección - Indica la dirección
secundaria de la nave
        PRECONDICIÓN:
            #Ninguna
    */
    IrAlBorde(opuesto(dirPrincipal))
    IrAlBorde(opuesto(dirSecundaria))
}

function estoyFinalDelRecorrido__(dirPrincipal, dirSecundaria){

    return(not puedeMover(dirPrincipal) && not
puedeMover(dirSecundaria))
```

```
}
```

```
procedure DarUnPasoHacia__(dirPrincipal, dirSecundaria){
    if(puedeMover(dirPrincipal)){
        Mover(dirPrincipal)
    }else{
        IrAlBorde(opuesto(dirPrincipal))
        Mover(dirSecundaria)
    }
}
```

Comentario:

El nombre contarSiHayTripulante no es adecuado, pues las funciones deberían ser nombradas con sustantivos.

Los procedimientos IrAlIncioDeLaNave__, DarUnPasoHacia__ y la función estoyFinalDelRecorrido__ son análogas a las realizadas en la práctica para recorrer el tablero, por lo que podías reutilizar las ya realizadas.

Cuando invocas a la función estoyAlFinalDeLaNave faltan los guiones bajos al final, que si están en la definición.

En contarSiHayTripulante, la condición luego del when debería llevar paréntesis (por sintaxis).

3)Escribir la función habráUnaExpulsiónInjusta que indica si los jugadores van a determinar expulsar a alguien de la nave injustamente. Esto ocurre cuando en cualquier lugar de la nave un tripulante está sobre una rejilla de ventilación.

Codigo:

```
function habraUnaExpulsionInjusta(numeroDeHabitacion){
/*PROPOSITO:indica si los jugadores van a determinar expulsar a alguien de la nave injustamente a un tripulante en cualquier lugar de la nave.
PRECONDICION:no hay
RESULTADO:Booleano
OBSERVACION:Esto ocurre cuando en cualquier lugar de la nave un tripulante está sobre una rejilla de ventilación.
```

```
*/
    IrAHabitaciónNúmero_(númeroDeHabitación)

    while (not hayTripulanteSobreRejilla()){
        hayTripulanteSobreRejilla()
        IrAPróximaZonaEnLaHabitaciónAl_Y_(Este,Norte)
    }
    return(hayTripulanteSobreRejilla())

}

function hayTripulanteSobreRejilla(){
/*PROPOSITO:indica si hay un tripulante sobre la rejilla
PRECONDICION:debe haber un tripulante en la zona
RESULTADO:bool
*/

    return(hayTripulante() && hayRejilla())
}

function hayRejilla(){
/*PROPOSITO:indica si hay una rejilla en la zona
PRECONDICION:no hay
RESULTADO:booleano
*/
    return(hayBolitas(colorRejilla())
}
```

Comentario:

Dentro del while estás utilizando la función "hayTripulanteSobreRejilla" como un comando, cuando es una expresión, y solo puede usarse como tal(preguntar esto ultimo). A esto se suma que estas mezclando un recorrido de habitaciones con un recorrido sobre las zonas de la habitación, no terminando por recorrer la totalidad de la nave.

4) Escribir el procedimiento AcercarseParaMatar que hace que el impostor se mueva una posición en forma ortogonal (a una de las zonas lindantes) con la intención de acercarse al impostor que está cerca, para matarlo posteriormente. Para esto se presupone que en la zona actual hay un impostor y que en alguna de las celdas en diagonal a la actual (hacia cualquiera de las 4 diagonales) hay un tripulante, y que estos jugadores son los únicos dos en la habitación. Así, por ej. si el tripulante está en la diagonal Norte-Este, entonces el impostor debe moverse a la celda al Norte o al Este.

```
procedure AcercarseParaMatar(){

/*PROPOSITO:hace que el impostor se mueva una posición en forma
ortogonal (a una de las zonas lindantes) con la intención de
acercarse al impostor

que está cerca, para matarlo posteriormente

PRECONDICION:

*en la zona actual hay un el impostor y en las zonas diagonales
hay un tripulante.

*solamente en la habitacion hay un impostor e tripulante.

*/

    IrAlPrincipioDeLaHabitaciónAl_Y_(Este,Norte)

    while(not hayImpostor()){

        MatarTripulanteSiHay()

        MoverImpostor()

    }


}


procedure MatarTripulanteSiHay(){

/*PROPOSITO:El impostor mata al tripulante

PRECONDICION:no hay

OBSERVACION:hayImpostor y hayTripulante se consideran
implementadas en ejercicios anteriores

*/

    if (hayImpostor() && hayTripulante()){


        Sacar(colorJugador())


    }
```

```
}

procedure MoverImpostorADireccion__(dir1,dir2){
/*PRECONDICION:El impostor se mueve de una zona a otra
PRECONDICION:debe haber un impostor en la celda actual
dir1:direccion en fila
dir2:direccion en columna
*/
    Sacar(colorImpostor())
    IrAPróximaZonaEnLaHabitaciónAl_Y_(dir1, dir2)
    Poner(colorImpostor())
}
```

Comentario:

Lo realizado no condice con lo pedido. No había que buscar a los jugadores por toda la habitación, sino que estaba garantizado que había un impostor en la zona actual, y que el tripulante estaba en alguna de las diagonales.

En tu código haces uso de MoverImpostor, pero nunca implementaste tal procedimiento, sino que lo que implementaste es MoverImpostorADireccion__. Allí no terminas de implementar correctamente lo que planteas, pues un impostor no está dado unicamente por una bolita de dicho color, sino también por bolitas del color jugador.

5) Se quiere saber si en algún lado hay más de un impostor agazapado esperando una víctima, es decir, si en una habitación hay 2 o más impostores y ningún tripulante. Se pide, entonces implementar la función hayImpostoresAgazapadosEnAlgunaHabitación que indica lo mencionado.

```
function
hayImpostoresAgazapadosEnAlgunaHabitación(nroHabitacion){
/*PROPOSITO:indica si hay 2 o mas impostores en una habitacion
PRECONDICION:No debe haber tripulantes en la habitacion
RESULTADO:Booleano
nroHabitacion:es el numero de la habitacion a verificar
*/

    IrAHabitaciónNúmero_(nroDeHabitación)
```

```
    cantidadDeImpostoresVistos:=0
    IrAlPrincipioDeLaHabitaciónAl_Y_(Este,Norte)
    while (not hayPróximaZonaEnLaHabitaciónAl_Y_(Este,Norte)){
        cantidadDeImpostoresVistos:=cantidadDeImpostoresVistos +
contarImpostores()
        IrAPróximaZonaEnLaHabitaciónAl_Y_(Este,Norte)
    }

    return(cantidadDeImpostoresVistos + contarImpostores()>=2)

}

function contarImpostores(){
/*PROPOSITO:Cuenta 1 si hay impostores 0 si no hay
PRECONDICION:no hay
RESULTADO:numero
*/

    return(choose 1 when hayImpostor()
                  0 otherwise)
}
```

Comentario:

En primer lugar mencionar que "hayImpostoresAgazapadosEnAlgunaHabitación" no esperaba parámetros de ningun tipo, por lo que estás simplificando el problema al limitarlo a una única habitación agregando un parámetro que no estaba dado.

Notar además que hayImpostoresAgazapadosEnAlgunaHabitación requería no solo 2 o más impostores, sino asegurarse también que no haya "nadie más", es decir, que no hubiera tripulantes, algo que tu código no está contemplando.

Incluso así, mencionar que una mejor estrategia hubiera sido separar el hecho de contar del comparar, optando por una solución que incluya recaer en una subtarea como "cantidadDeImpostoresEnLaHabitación() >= 2" donde cantidadDeImpostoresEnLaHabitación es quien se encarga de recorrer y contar los impostores de la habitación. Más aún mencionar que "contarImpostores" no es un buen nombre, pues las funciones deberían estar nombradas con sustantivos, no verbos, pues describen datos, no acciones. Por último, detalle, la condición luego del when debe estar dentro de paréntesis, por sintaxis.