//================================Biblioteca================================//

//================================Movimiento================================//

procedure Mover\_VecesAl\_(cantidadAMover, direcciónAMover){

/\* Prop. Mover el cabezal \*cantidadAMover\* celdas hacia el \*direcciónAMover\*.

Prec. Debe haber \*cantidadAMover\* celdas hacia el \*direcciónAMover\* del cabezal

Param. \*cantidadAMover\*. Numero. La cantidad de celdas que se moverá el cabezal. \*direcciónAMover\* . Dirección. La dirección en la que se va a mover el cabezal.

\*/

repeat(cantidadAMover){

Mover(direcciónAMover)

}

}

procedure Mover\_Si\_(dirección, condición){ //Mover Condicional

/\* Prop. Mover hacia el \*dirección\* solamente si \*condición\* es verdadera.

Prec. Debe haber 1 celda al \*dirección\*

Param. \*dirección\*. Dirección. Dirección a la que se moverá. \*/

if(condición){

Mover(dirección)

}

}

//================================Poner/Sacar================================//

procedure Poner\_DeColor\_(cantidadAPoner, colorAPoner){

/\* Prop. Poner \*cantidad\* de color \*color\* en la celda actual.

Prec. Ninguna

Param. \*cantidadAPoner\*. Numero. La cantidad de bolitas a poner

\*colorAPoner\* . Color. El color de la(s) bolitas a poner. \*/

repeat(cantidadAPoner){

Poner(colorAPoner)

}

}

procedure Sacar\_DeColor\_(cantidadASacar, colorASacar){

/\* Prop. Sacar \*cantidadASacar\* de color \*colorASacar\* en la celda actual.

Prec. Debe haber al menos \*cantidadASacar\* de \*colorASacar\*

Param. \*cantidadASacar\*. Numero. La cantidad de bolitas a sacar.

\*colorASacar\*. Color. El color de la(s) bolitas que se van a sacar. \*/

repeat(cantidadASacar){

Sacar(colorASacar)

}

}

procedure PonerUnaDeCada(){

/\* Propósito: Poner una bolita de cada color en la celda actual.

Precondición: Ninguna. \*/

Poner(Azul)

Poner(Negro)

Poner(Rojo)

Poner(Verde)

}

procedure SacarTodasLasDeColor\_(colorASacar){

/\*Prop. Sacar todas las bolitas \*colorASacar\* del la celda actual.

Prec. Ninguna.

Param. \*colorASacar\*. Color. El color de la/s bolitas a sacar \*/

Sacar\_DeColor\_(nroBolitas(colorASacar), colorASacar)

}

procedure VaciarCelda(){

/\* Prop. Sacar todas las bolitas de todos los colores de la celda actual.

Prec. Ninguna. \*/

Sacar\_DeColor\_(nroBolitas(Rojo), Rojo)

Sacar\_DeColor\_(nroBolitas(Verde), Verde)

Sacar\_DeColor\_(nroBolitas(Negro), Negro)

Sacar\_DeColor\_(nroBolitas(Azul), Azul)

}

procedure Poner\_Si\_(color, condición){ //Poner con condicional

/\* Prop. Poner una bolita de \*color\* solamente si \*condición\* es verdadera.

Prec. Ninguna

Param.\*color\*. Color. Color de la bolita a poner.

\*condición\*. Booleano. La condición que se debe cumplir \*/

if(condición){

Poner(color)

}

}

procedure Sacar\_Si\_(color, condición){ //Sacar con condicional

/\* Prop. Sacar una bolita de \*color\* solamente si \*condición\* es verdadera.

Prec. Debe haber al menos una bolita de color \*color\*

Param. \*color\*. Color. Color de la bolita a poner.\*/

if(condición){

Sacar(color)

}

}

//==========Recorridos==========//

procedure IrAPrimeraCeldaEnUnRecorridoAl\_Y\_(dirPrincipal, dirSecundaria) {

/\* Prop. Ir al inicio de un recorrido hacia \*dirPrincipal\* - \*dirSecundaria\*

Prec. \*dirPrincipal\* y \*dirSecundaria\* NO DEBEN ser ni iguales ni opuestos

Param. \*dirPrincipal\*. Dirección. Primer dirección del recorrido.

\*dirSecundaria\*. Dirección. Segunda dirección del recorrido\*/

IrAlBorde(opuesto(dirPrincipal))

IrAlBorde(opuesto(dirSecundaria))

}

function haySiguienteCeldaEnUnRecorridoAl\_Y\_(dirPrincipal, dirSecundaria){

/\* Propósito: Indica si hay una celda valida en el recorrido.

Precondiciones: Ninguna:

Parámetros: \*dirPrincipal\*: Dirección - Dirección principal del recorrido. \*dirSecundaria\*: Dirección - Dirección secundaria del recorrido.

Tipo: Booleano.\*/

return(

puedeMover(dirPrincipal)||

puedeMover(dirSecundaria)

)

}

procedure IrASiguienteCeldaEnUnRecorridoAl\_Y\_(dirPrincipal, dirSecundaria){

/\* Prop. Ir a la siguiente celda de un recorrido hacia el \*dirPrincipal\* - \*dirSecundaria\*.

Prec.NingunaParam.\*dirPrincipal\*. Dirección. Primer dirección del recorrido.

\*dirSecundaria\*. Dirección. Segunda dirección del recorrido. \*/

if(puedeMover(dirPrincipal)){

Mover(dirPrincipal)

}else{

IrAlBorde(opuesto(dirPrincipal))

Mover(dirSecundaria)

}

}

//==========Funciones Booleanas=========//

function tieneUnaDeCada(){

/\* Prop. Indica si hay al menos una bolita de cada color en la celda actual.

Prec. Ninguna

Tipo: Booleano \*/

return(

hayBolitas(Negro) &&

hayBolitas(Rojo) &&

hayBolitas(Azul) &&

hayBolitas(Verde))

}

function esCeldaVacía(){

/\*Prop. Indica si la celda actual está vacía.

Prec. Ninguna

Tipo: Booleano \*/

return(

not hayBolitas(Negro) &&

not hayBolitas(Rojo) &&

not hayBolitas(Azul)&&

not hayBolitas(Verde))

}

function esCeldaConBolitas(){

/\* Prop. Indica si la celda actual no está vacía.

Prec. Ninguna

Tipo: Booleano \*/

return(

not esCeldaVacía()

)

}

function hayBolitas\_Al\_(color, dirección){

/\*Propósito: Indica si hay celda lindante \*dirección\* y si la misma tiene bolitas de color \*color\*

Prec: -

Parámetros: \*color\*: Color - El color de la bolita a verificar.

\*dirección\*: Dirección - Dirección de la celda a verificar.

Tipo:Booleano \*/

return(

puedeMover(dirección) &&

hayBolitas\_EnCeldaAl\_(color, dirección))

}

function mínimoEntre\_Y\_(valor1, valor2){

/\* Propósito: Describe el valor que sea mas chico.

Precondición: -

Parámetros: \*valor1\*: - El primer valor a comparar. \*valor2\*: - El segundo valor a comparar

Tipo: \*/

return (

choose

valor1 when (valor1 < valor2)

valor2 otherwise

)

}

function máximoEntre\_Y\_(valor1, valor2){

/\* Propósito: Describe el valor que sea mas grande.

Precondición: -

Parámetros: \*valor1\*: - El primer valor a comparar. \*valor2\*: - El segundo valor a comparar

Tipo: \*/

return (

choose

valor1 when (valor1 > valor2)

valor2 otherwise

)

}

function distanciaAlBorde\_(dirección){

/\* Propósito: Describe la cantidad de celdas entre la celda actual y el borde \*dirección\*.

Precondición: -

Parámetros: \*dirección\*: Dirección - La dirección del borde

Tipo: Número \*/

distancia := 0

while(puedeMover(dirección)){

Mover(dirección)

distancia := distancia + 1

}

return (distancia)

}

function coordenadaX(){

/\* Propósito: Indica la coordenada X.

Prec. Ninguna

Tipo: Número \*/

contadorX :=0

while(puedeMover(Oeste)){

contadorX:= contadorX + 1

Mover(Oeste)

}

return(

contadorX

)

}

function coordenadaY(){

/\* Propósito: Indica la coordenada Y.

Prec. Ninguna

Tipo: Número \*/

contadorY :=0

while(puedeMover(Sur)){

contadorY:= contadorY + 1

Mover(Sur)

}

return(

contadorY

)

}

function nroFilas(){

/\* Propósito: Describe la cantidad de filas del tablero.

Precondición: -

Tipo: Número \*/

IrAlBorde(Sur)

filas := 1

while(puedeMover(Norte)){

Mover(Norte)

filas := filas + 1

}

return (filas)

}

function nroColumnas(){

/\* Propósito: Describe la cantidad de filas del tablero.

Precondición: -

Tipo: Número \*/

IrAlBorde(Oeste)

columnas := 1

while(puedeMover(Este)){

Mover(Este)

columnas := columnas + 1

}

return (columnas)

}

function unoSi\_CeroSino(condición){

/\* Propósito: Describe 1 si la condición es verdadera ó 0 sino.

Precondición: -

Parámetros: \*condición\*: Booleano - La condición a evaluar.

Tipo: Número \*/

return(

choose

1 when (condición)

0 otherwise

)

}

function nroVacías(){

/\* Propósito: Describe la cantidad de Celdas vacías en el tablero

Precondición: Ninguna

Tipo: Número \*/

IrAPrimeraCeldaEnUnRecorridoAl\_Y\_(Este, Norte)

cantidadVacías:= 0 + unoSi\_CeroSino(esCeldaVacía())

while(haySiguienteCeldaEnUnRecorridoAl\_Y\_(Este, Norte)){

IrASiguienteCeldaEnUnRecorridoAl\_Y\_(Este, Norte)

cantidadVacías := cantidadVacías + unoSi\_CeroSino(esCeldaVacía())

}

return(cantidadVacías)

}

function cantidadDeCeldasConBolitasDeColor\_(color){

/\* Propósito: Describe la cantidad de Celdas con bolitas de color

\*color\* en el tablero

Precondición: Ninguna

Parámetros: \*color\*: Color - El color a buscar (?)

Tipo: Número \*/

IrAPrimeraCeldaEnUnRecorridoAl\_Y\_(Este, Norte)

celdasConBolitasDeColor\_:= 0 + unoSi\_CeroSino(nroBolitas(color) > 0)

while(haySiguienteCeldaEnUnRecorridoAl\_Y\_(Este, Norte)){

IrASiguienteCeldaEnUnRecorridoAl\_Y\_(Este, Norte)

celdasConBolitasDeColor\_:= celdasConBolitasDeColor\_ + unoSi\_CeroSino(nroBolitas(color) > 0)

}

return(celdasConBolitasDeColor\_)

}

function nroBolitasTotalDeColor\_(color){

/\* Propósito: Describe la cantidad de total de bolitas de color

\*color\* en el tablero

Precondición: Ninguna

Parámetros: \*color\*: Color - El color a buscar (?)

Tipo: Número \*/

IrAPrimeraCeldaEnUnRecorridoAl\_Y\_(Este, Norte)

totalBolitasDeColor\_:= 0 + nroBolitas(color)

while(haySiguienteCeldaEnUnRecorridoAl\_Y\_(Este, Norte)){

IrASiguienteCeldaEnUnRecorridoAl\_Y\_(Este, Norte)

totalBolitasDeColor\_:= totalBolitasDeColor\_ +nroBolitas(color)

}

return(totalBolitasDeColor\_)

}

//======================Recorrido Por Celdas====================//

IrAPrimeraCeldaEnUnRecorridoAl\_Y\_(Este, Norte) //Inicia recorrido

while(haySiguienteCeldaEnUnRecorridoAl\_Y\_(Este, Norte)){

ProcedimientoP() //Procesa

IrASiguienteCeldaEnUnRecorridoAl\_Y\_(Este, Norte) //Ir al siguiente elemento

}

//======================Recorrido Por Fila====================//

IrAlBorde(Oeste) //Inicia recorrido

while(puedeMover(Este)){

ProcedimientoP() //Procesa

Mover(Este) //Ir al siguiente elemento

}

//======================Recorrido Por Columna====================//

IrAlBorde(Sur) //Inicia recorrido

while(puedeMover(Norte)){

ProcedimientoP() //Procesa

Mover(Norte) //Ir al siguiente elemento

}

//======================Recorrido De Busqueda====================//

IrAPrimeraCeldaEnUnRecorridoAl\_Y\_(Este, Norte) //Inicia recorrido

while(not hayElementoABuscar){

IrASiguienteCeldaEnUnRecorridoAl\_Y\_(Este, Norte)

}

//======================Recorrido sobre enumerativo====================//  
 pepe := minDir() //

ProcedimientoP()

while(pepe /= maxDir()){

pepe := siguiente(pepe)

ProcedimientoP()

}

//======================Recorrido sobre enumerativo====================//  
 pepe :=minColor()

ProcedimientoP()

while(pepe /= maxColor()){

pepe := siguiente(pepe)

ProcedimientoP()

}

//======================Recorrido sobre enumerativo====================//

IrAPrimeraCeldaEnUnRecorridoAl\_Y\_(Este, Norte)

variableX := nroBolitas(Rojo) // o X valor

while(haySiguienteCeldaEnUnRecorridoAl\_Y\_(Este, Norte)){

IrASiguienteCeldaEnUnRecorridoAl\_Y\_(Este, Norte)

variableX := variableX + nroBolitas(Rojo) // o + unoSi\_CeroSino()

}

return(variableX)