***PRÁCTICA 9***

# ***// EJ 1 (HECHO)***

**function hayBolitas\_EnCeldaAl\_(color, dirección){**

**/\***

**Propósito: Indica si la celda lindante en la dirección \*\*dirección\*\* tiene bolitas de color \*\*color\*\*.**

**Parámetros:**

**\* color: Color - El color de las bolitas a ver si hay.**

**\* dirección: Dirección - La dirección en la que se ve si hay bolitas.**

**Precondición: Debe haber una celda lindante en la dirección \*\*dirección\*\***

**OBS. : Si no hay celda lindante en la dirección \*\*dirección\*\*, hace BOOM.**

**Tipo: Booleano.**

**\*/**

**Mover(dirección)**

**return (hayBolitas(color))**

**}**

# ***// EJ 2 (HECHO)***

## **BIBLIOTECA**

**function tieneBolitas\_Al\_(color, dirección){**

**/\***

**Propósito: Indica si la celda lindante en la dirección \*\*dirección\*\* tiene bolitas de color \*\*color\*\*.**

**Parámetros:**

**\* color: Color - El color de las bolitas a ver si hay.**

**\* dirección: Dirección - La dirección en la que se ve si hay bolitas.**

**Precondición: Debe haber una celda lindante en la dirección \*\*dirección\*\*.**

**Tipo: Booleano.**

**\*/**

**Mover(dirección)**

**return (hayBolitas(color))**

**}**

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

**/\***

**Propósito: Indica si hay una celda lindante en la dirección \*\*dirección\*\* y la misma tiene**

**bolitas de color \*\*color\*\*.**

**Parámetros:**

**\* color: Color - El color de las bolitas a ver si hay.**

**\* dirección: Dirección - La dirección en la que se ve si hay una celda.**

**Precondición: Ninguna.**

**Tipo: Booleano.**

**\*/**

**return(puedeMover(dirección) && tieneBolitas\_Al\_(color, dirección))**

**}**

# ***// EJ 3(HECHO)***

**function hayBolitas\_EnElBorde\_(color, dirección){**

**/\***

**Propósito: Indica si la celda que se encuentra en el borde en la dirección \*\*dirección\*\***

**tiene bolitas de color \*\*color\*\*.**

**Parámetros:**

**\* color: Color - El color de las bolitas a ver si hay.**

**\* dirección: Dirección - La dirección en la que se ve si hay bolitas.**

**Precondición: Ninguna**

**Tipo: Booleano.**

**\*/**

**IrAlBorde(dirección)**

**return (hayBolitas(color))**

**}**

# ***// EJ 4 (HECHO)***

**function hayBolitas\_Hacia\_(color, dirección){**

**/\***

**Propósito: Indica si alguna de las celdas hacia la dirección \*\*dirección\*\***

**tiene bolitas de color \*\*color\*\*.**

**Parámetros:**

**\* color: Color - El color de las bolitas a ver si hay.**

**\* dirección: Dirección - La dirección en la que se ve si hay bolitas.**

**Precondición: Ninguna**

**Tipo: Booleano.**

**\*/**

**while(puedeMover(dirección)){**

**Mover(dirección)**

**}**

**return (hayBolitas(color))**

**}**

# ***// EJ 5 (HECHO)***

**function hayCeldaVacíaHacia\_(dirección){**

**/\***

**Propósito: Indica si hay alguna celda vacía hacia la dirección \*\*dirección\*\*.**

**Parámetros:**

**\* dirección: Dirección - La dirección en la que se ve si hay celda vacia.**

**Precondición: Debe haber una celda lindante en la dirección \*\*dirección\*\***

**Tipo: Booleano.**

**\*/**

**while(puedeMover(dirección)){**

**Mover(dirección)**

**}**

**return (esCeldaVacía())**

**}**

# ***// EJ 6 (HECHO)***

**function hayAlgunaBolita\_(color){**

**/\***

**Propósito: Indica si hay alguna bolita de color \*\*color\*\* en el tablero.**

**Parámetros:**

**\* color: Color - El color de las bolitas a ver si hay.**

**Precondición: Ninguna.**

**Tipo: Booleano.**

**\*/**

**IrAPrimeraCeldaEnUnRecorridoAl\_Y\_(Norte, Este)**

**while(haySiguienteCeldaEnUnRecorridoAl\_Y\_(Norte, Este) && esCeldaVacía()){**

**IrASiguienteCeldaEnUnRecorridoAl\_Y\_(Norte, Este)**

**}**

**return (hayBolitas(color))**

**}**

# ***// EJ 7 (HECHO)***

**function hayAlgunaCeldaVacía(){**

**/\***

**Propósito: Indica si hay alguna celda vacia en el tablero.**

**Precondición: Ninguna.**

**Tipo: Booleano.**

**\*/**

**IrAPrimeraCeldaEnUnRecorridoAl\_Y\_(Norte, Este)**

**while(haySiguienteCeldaEnUnRecorridoAl\_Y\_(Norte, Este) && not(esCeldaVacía())){**

**IrASiguienteCeldaEnUnRecorridoAl\_Y\_(Norte, Este)**

**}**

**return (esCeldaVacía())**

**}**

# ***// EJ 8***

# ***// EJ 9***

# ***// EJ 10***

# ***// EJ 11 (HECHO)***

## **BIBLIOTECA**

**function mínimoEntre\_Y\_(primerValor, segundoValor){**

**/\***

**Propósito: Describe el valor más chico entre el valor \*\*primerValor\*\* y \*\*segundoValor\*\*.**

**Parámetros:**

**\* primerValor: Número ó Color ó Dirección - El primer valor a comparar.**

**\* segundoValor: Número ó Color ó Dirección - El segundo valor a comparar.**

**Precondición: Los valores deben ser del mismo tipo y diferentes uno del otro.**

**Tipo: Número ó Color ó Dirección.**

**\*/**

**return(**

**choose**

**primerValor when (primerValor<segundoValor)**

**segundoValor otherwise**

**)**

**}**

# ***// EJ 12 (HECHO)***

## **BIBLIOTECA**

**function máximoEntre\_Y\_(primerValor, segundoValor){**

**/\***

**Propósito: Describe el valor más grande entre el valor \*\*primerValor\*\* y \*\*segundoValor\*\*.**

**Parámetros:**

**\* primerValor: Número ó Color ó Dirección - El primer valor a comparar.**

**\* segundoValor: Número ó Color ó Dirección - El segundo valor a comparar.**

**Precondición: Los valores deben ser del mismo tipo y diferentes uno del otro.**

**Tipo: Número ó Color ó Dirección.**

**\*/**

**return(**

**choose**

**primerValor when (primerValor>segundoValor)**

**segundoValor otherwise**

**)**

**}**

# ***// EJ 13***

# ***// EJ 14***

# ***// EJ 15***

# ***// EJ 16***

# ***// EJ 17 (HECHO)***

## **BIBLIOTECA**

**function distanciaAlBorde\_(dirección){**

**/\***

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

**Parámetros:**

**\* dirección: Dirección - La dirección del borde a saber la distancia.**

**Precondición: Ninguna.**

**Tipo: Número.**

**Observaciones: Si la celda actual es la del borde \*\*dirección\*\* la distancia es 0.**

**\*/**

**distancia := 0**

**while(puedeMover(dirección)){**

**Mover(dirección)**

**distancia := distancia+1**

**}**

**return(distancia)**

**}**

# ***// EJ 18 (HECHO)***

## **BIBLIOTECA**

**function coordenadaX(){**

**/\***

**Propósito: Describe la coordenada de la columna de la celda actual.**

**Precondición: Ninguna.**

**Tipo: Número.**

**\*/**

**return(distanciaAlBorde\_(Oeste))**

**}**

**function coordenadaY(){**

**/\***

**Propósito: Describe la coordenada de la fila de la celda actual.**

**Precondición: Ninguna.**

**Tipo: Número.**

**\*/**

**return(distanciaAlBorde\_(Sur))**

**}**

# ***// EJ 19 (HECHO)***

## **BIBLIOTECA**

**function nroFilas(){**

**/\***

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

**Precondición: Ninguna.**

**Tipo: Número.**

**\*/**

**return(distanciaAlBorde\_(Sur)+distanciaAlBorde\_(Norte)+1)**

**}**

**function nroColumnas(){**

**/\***

**Propósito: Describe la cantidad de columnas del tablero.**

**Precondición: Ninguna.**

**Tipo: Número.**

**\*/**

**return(distanciaAlBorde\_(Oeste)+distanciaAlBorde\_(Este)+1)**

**}**

# ***// EJ 20 (HECHO)***

## **BIBLIOTECA**

**function nroVacías(){**

**/\***

**Propósito: Describe la cantidad de celdas vacías que hay en el tablero.**

**Precondición: Ninguna.**

**Tipo: Número.**

**\*/**

**IrAPrimeraCeldaEnUnRecorridoAl\_Y\_(Este, Norte)**

**cantidadDeCeldasVacíasVistas := unoSi\_CeroSino(esCeldaVacía())**

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

**IrASiguienteCeldaEnUnRecorridoAl\_Y\_(Este, Norte)**

**cantidadDeCeldasVacíasVistas := cantidadDeCeldasVacíasVistas+unoSi\_CeroSino(esCeldaVacía())**

**}**

**return(cantidadDeCeldasVacíasVistas)**

**}**

# ***// EJ 21 (HECHO)***

## **BIBLIOTECA**

**function cantidadDeCeldasConBolitasDeColor\_(color){**

**/\***

**Propósito: Describe la cantidad de celdas que contengan al menos una bolita de color \*\*color\*\*.**

**Precondición: Ninguna.**

**Tipo: Número.**

**\*/**

**IrAPrimeraCeldaEnUnRecorridoAl\_Y\_(Este, Norte)**

**cantidadDeCeldasConBolitasDeColorDado := unoSi\_CeroSino(hayBolitas(color))**

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

**IrASiguienteCeldaEnUnRecorridoAl\_Y\_(Este, Norte)**

**cantidadDeCeldasConBolitasDeColorDado := cantidadDeCeldasConBolitasDeColorDado+unoSi\_CeroSino(hayBolitas(color))**

**}**

**return(cantidadDeCeldasConBolitasDeColorDado)**

**}**

# ***// EJ 22 (HECHO)***

## **BIBLIOTECA**

**function nroBolitasTotalDeColor\_(color){**

**/\***

**Propósito: Describe la cantidad de bolitas de color \*\*color\*\* que hay en el tablero.**

**Precondición: Ninguna.**

**Tipo: Número.**

**\*/**

**IrAPrimeraCeldaEnUnRecorridoAl\_Y\_(Este, Norte)**

**cantidadDeBolitasDeColorDado := nroBolitas(color)**

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

**IrASiguienteCeldaEnUnRecorridoAl\_Y\_(Este, Norte)**

**cantidadDeBolitasDeColorDado := cantidadDeBolitasDeColorDado+nroBolitas(color)**

**}**

**return(cantidadDeBolitasDeColorDado)**

**}**

# ***// EJ 23 (HECHO)***

**function cantidadDeAutosEnLaCarretera(){**

**/\***

**PROPÓSITO: Describe la cantidad de autos en la carretera.**

**TIPO: Numero.**

**PRECONDICIONES: Ninguna.**

**\*/**

**IrAPrimeraCeldaEnUnRecorridoAl\_Y\_(Norte, Este)**

**cantidadAutos := nroBolitas(Rojo)**

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

**IrASiguienteCeldaEnUnRecorridoAl\_Y\_(Norte, Este)**

**cantidadAutos := cantidadAutos + nroBolitas(Rojo)**

**}**

**return(cantidadAutos)**

**}**

**function hayUnAuto(){**

**/\***

**PROPÓSITO: Indica si hay un auto en la celda actual.**

**TIPO: Booleano.**

**PRECONDICIONES: Ninguna.**

**\*/**

**return(hayBolitas(Rojo))**

**}**

**program {**

**return(cantidadDeAutosEnLaCarretera())**

**}**

# ***// EJ 24 (HECHO)***

**function cantidadTotalDeÁrbolesEnElTerreno(){**

**/\***

**PROPOSITO: Indica la cantidad de árboles total que hay en el terreno.**

**PRECONDICION: Ninguna.**

**TIPO: Número.**

**\*/**

**IrAPrimeraCeldaEnUnRecorridoAl\_Y\_(Este, Norte)**

**cantidadÁrbolesTotal := cantidadÁrboles()**

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

**IrASiguienteCeldaEnUnRecorridoAl\_Y\_(Este, Norte)**

**cantidadÁrbolesTotal := cantidadÁrbolesTotal + cantidadÁrboles()**

**}**

**return(cantidadÁrbolesTotal)**

**}**

**function cantidadTotalDeÁrbolesEnElTerrenoLuegoDeExplociones(){**

**/\***

**PROPOSITO: Indica la cantidad de árboles total que hubiera en el terreno luego de**

**que explotaran todas las bombas del mismo.**

**PRECONDICION: Ninguna.**

**TIPO: Número.**

**\*/**

**ExplotarTodasLasBombasDelTerreno()**

**IrAPrimeraCeldaEnUnRecorridoAl\_Y\_(Este, Norte)**

**cantidadÁrbolesTrasBombas := cantidadÁrboles()**

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

**IrASiguienteCeldaEnUnRecorridoAl\_Y\_(Este, Norte)**

**cantidadÁrbolesTrasBombas := cantidadÁrbolesTrasBombas + cantidadÁrboles()**

**}**

**return(cantidadÁrbolesTrasBombas)**

**}**

**procedure ExplotarTodasLasBombasDelTerreno(){**

**/\***

**PROPOSITO: Explota todas las bombas que haya en el terreno.**

**PRECONDICION: Ninguna.**

**\*/**

**IrAPrimeraCeldaEnUnRecorridoAl\_Y\_(Este, Norte)**

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

**ExplotarTodasLasBombasAcá()**

**IrASiguienteCeldaEnUnRecorridoAl\_Y\_(Este, Norte)**

**}**

**ExplotarTodasLasBombasAcá()**

**}**

**procedure ExplotarTodasLasBombasAcá(){**

**/\***

**PROPOSITO: Explota todas las bombas de la ubicación actual.**

**PRECONDICION: Ninguna.**

**\*/**

**while(hayBombaAcá()){**

**ExplotarBomba()**

**}**

**}**

# ***// EJ 25 (HECHO)***

## **BIBLIOTECA**

**function unoSi\_CeroSino(condición){**

**/\***

**Propósito: Describe 1 si se cumple la condición \*\*condición\*\*, si no se cumple describe 0.**

**Parámetros:**

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

**Precondición: Ninguna.**

**Tipo: Número.**

**\*/**

**return(**

**choose**

**1 when (condición)**

**0 otherwise**

**)**

**}**

# ***// EJ 26***

# ***// EJ 27 (HECHO)***

**program {**

**PonerUnaDeCadaColor()**

**}**

**procedure PonerUnaDeCadaColor() {**

**/\***

**Prop.:**

**Pone un bolita de cada color en la celda actual**

**Prec.:**

**Ninguna**

**\*/**

**colorRecorrido := minColor()**

**Poner(colorRecorrido)**

**while (colorRecorrido /= maxColor()) {**

**colorRecorrido := siguiente(colorRecorrido)**

**Poner(colorRecorrido)**

**}**

**}**

# ***// EJ 28 (HECHO)***

**procedure LimpiarCruzDeColor\_(color) {**

**/\***

**Propósito: Limpia el dibujo de una cruz realizado con bolitas**

**de color \*\*color\*\*.**

**Parámetros:**

**\* color: Color - El color de las bolitas a ver si hay.**

**Precondición: El cabezal debe estar ubicado en el centro de la cruz.**

**Tipo: Número.**

**\*/**

**SacarTodasLasDeColor\_(color)**

**direccion := minDir()**

**while(direccion /= maxDir()){**

**SacarTodasLas\_Hacia\_(color, direccion)**

**direccion := siguiente(direccion)**

**}**

**SacarTodasLas\_Hacia\_(color, direccion)**

**}**

**procedure SacarTodasLasDeColor\_(color) {**

**/\***

**Prop.: Quitar todas las bolitas del color \*color\* en la celda actual**

**Prec.: Ninguna**

**Param: \*color\* - Color: Indica el color de las bolitas a sacar**

**\*/**

**while(hayBolitas(color)) {**

**Sacar(color)**

**}**

**}**

**procedure SacarTodasLas\_Hacia\_(color, direccion) {**

**/\***

**Prop.: Quitar todas las bolitas del color \*color\* en la celda actual**

**Prec.: Ninguna**

**Param:**

**\* color: Color - El color de las bolitas a ver si hay.**

**\* dirección: Dirección - La dirección en la que se ve si hay bolitas.**

**\*/**

**pasosAvanzados := 0**

**while(puedeMover(direccion)) {**

**Mover(direccion)**

**SacarTodasLasDeColor\_(color)**

**pasosAvanzados := pasosAvanzados + 1**

**}**

**Mover\_VecesAl\_(pasosAvanzados, opuesto(direccion))**

**}**

**program {**

**LimpiarCruzDeColor\_(Rojo)**

**}**

# ***// EJ 29***

# ***// EJ 30***

# ***// EJ 31***

# ***// EJ 32***

# ***// EJ 33***

# ***// EJ 34***

# ***// EJ 35***

# ***// EJ 36***

# 