You are given a chessboard with several rooks and bishops placed on some of its squares. How many unoccupied squares are there that are not under attack of any chess piece? Here, the standard rules are applied: a square is under attack of a rook or a bishop only if all squares between the piece and the current square are unoccupied.

* **[time limit] 3000ms (cs)**
* **[input] array.array.integer a**

matrix of size 8x8 containing numbers {-1, 0, 1} which represents chess pieces placement: -1 - bishop, 0 - empty square, 1 - rook

* **[output] integer**

number of safe squares on the board

<https://codefights.com/challenge/yYjAFYeG2hrWAjq4w/main>

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace ConsoleApplication1

{

class Program

{

static int bishopsAndRooks(int[][] a)

{

//List<int> filas = new List<int>();

//List<int> cols = new List<int>();

for (int i = 0; i < a.Length; i++)

{

for (int j = 0; j < a[i].Length; j++)

{

if (a[i][j] == 1)

{

int f = i, c = j;

//NORTE

while (f >= 0)

{

f--;

if (f >= 0 && (a[f][c] == 1 || a[f][c] == -1))

{

break;

}

else if (f >= 0 )

{

a[f][c] = 3;

}

}

//ESTE

f = i;

c = j;

while ( c < a[i].Length)

{

c++;

if ( c < a[i].Length && (a[f][c] == 1 || a[f][c] == -1))

{

break;

}

else if ( c < a[i].Length)

{

a[f][c] = 3;

}

}

//SUR

f = i; c = j;

while (f < a.Length)

{

f++;

if (f < a.Length && (a[f][c] == 1 || a[f][c] == -1))

{

break;

}

else if (f < a.Length )

{

a[f][c] = 3;

}

}

//OESTE

f = i; c = j;

while ( c >= 0)

{

c--;

if ( c >= 0 && (a[f][c] == 1 || a[f][c] == -1))

{

break;

}

else if ( c >= 0)

{

a[f][c] = 3;

}

}

}

else if (a[i][j] == -1)

{

int f = i, c = j;

//NE

while (f >= 0 && c < a[i].Length)

{

f--;

c++;

if (f >= 0 && c < a[i].Length && (a[f][c] == 1 || a[f][c] == -1))

{

break;

}

else if (f >= 0 && c < a[i].Length)

{

a[f][c] = 2;

}

}

f = i; c = j;

//SE

while (f < a.Length && c < a[i].Length)

{

f++;

c++;

if (f < a.Length && c < a[i].Length && (a[f][c] == 1 || a[f][c] == -1))

{

break;

}

else if (f < a.Length && c < a[i].Length)

{

a[f][c] = 2;

}

}

//SO

f = i; c = j;

while (f < a.Length && c >= 0)

{

f++;

c--;

if (f < a.Length && c >= 0 && (a[f][c] == 1 || a[f][c] == -1))

{

break;

}

else if (f < a.Length && c >= 0)

{

a[f][c] = 2;

}

}

//NO

f = i; c = j;

while (f >= 0 && c >= 0)

{

f--;

c--;

if (f >= 0 && c >= 0 && (a[f][c] == 1 || a[f][c] == -1))

{

break;

}

else if (f >= 0 && c >= 0)

{

a[f][c] = 2;

}

}

}

}

}

//for (int i = 0; i < a.Length; i++)

//{

// for (int j = 0; j < a[i].Length; j++)

// {

// if (filas.Contains(i) && a[i][j] == 0)

// {

// a[i][j] = 3;

// }

// if (cols.Contains(j) && a[i][j] == 0)

// {

// a[i][j] = 3;

// }

// }

//}

int ans = 0;

for (int i = 0; i < a.Length; i++)

{

for (int j = 0; j < a[i].Length; j++)

{

if (a[i][j] == -1)

{

Console.Write(" ");

}

else

{

Console.Write(" ");

}

Console.Write(a[i][j]);

if (a[i][j] == 0)

{

ans++;

}

}

Console.WriteLine();

}

return ans;

}

static void Main(string[] args)

{

int[][] matriz = {

new int[]{1,0,0,0,0,0,0,0},

new int[]{0,0,0,0,0,0,0,0},

new int[]{0,-1,0,0,1,0,0,0},

new int[]{0,0,0,0,0,0,0,0},

new int[]{0,0,0,0,0,0,0,0},

new int[]{0,0,0,-1,-1,0,0,0},

new int[]{0,0,0,0,0,0,0,0},

new int[]{0,0,0,0,0,0,0,0}};

//int[][] matriz = {

//new int[]{0,0,0,0,0,0,0,0},

//new int[]{0,0,0,0,0,0,0,0},

//new int[]{0,-1,0,0,0,0,0,0},

//new int[]{0,0,0,0,0,0,0,0},

//new int[]{0,0,0,0,0,0,0,0},

//new int[]{0,0,0,-1,-1,0,0,0},

//new int[]{0,0,0,0,0,0,0,0},

//new int[]{0,0,0,0,0,0,0,0}};

//int[][] matriz = {

// new int[]{0,0,0,0,0,1,0,0},

// new int[]{0,0,0,0,0,0,0,0},

// new int[]{0,0,0,0,0,0,0,0},

// new int[]{0,-1,1,0,1,0,0,0},

// new int[]{0,0,0,0,0,-1,0,0},

// new int[]{0,0,0,0,0,0,0,0},

// new int[]{0,0,0,1,0,0,0,0},

// new int[]{-1,0,0,0,0,0,0,0}};

Console.WriteLine( bishopsAndRooks(matriz) );

Console.ReadLine();

}

}

}