Day 1 – Task 2

Piles of books

After reading the input data, we initialize auxiliary matrix in which will we store information whether or not a pile is visible from some side (default value is 0 – false). For every row and column we determine number of visible piles by comparing it to the previous maximal value for the row and column. If the pile is higher than the current maximum then we set its value in the control matrix to 1 (visible) and update the maximum for that row (column). We repeat this procedure for all four directions.
Finally, we go through the control matrix and sum all the values, effectively counting the number of visible piles.

Another possible solution is to count all visible piles using following fact: pile is visible if and only if the height of current pile is strictly greater than all piles in some of four directions. So, we have to check whether every pile on left with same x coordinate is smaller than current pile. If not, then we must check every pile on right with same x coordinates. Also, apply the same for search for vertical direction.

Example solution in Pascal:

```
program Books;
```

Day 1 – Task 2

```
const
  MaxN = 55;
var
  mark: array [1.. MaxN, 1.. MaxN] of Boolean;
  b: array [1.. MaxN, 1.. MaxN] of LongInt;
  n, sol, i, j, max1, max2: LongInt;
begin

  // Input data
  ReadLn (n);
  for i := 1 to n do
    for j := 1 to n do
      Read (b [i][j]);
  sol := 0;
  FillChar (mark, SizeOf (mark), false);

  // mark [i][j] = True iff we can see books on field [i][j] from any direction
  // Find horizontal visible books
  for i := 1 to n do begin
    max1 := 0;
    max2 := 0;
    for j := 1 to n do begin
      // Visible from left to right
      if (b [i][j] > max1) then begin
        mark [i][j] := True;
        max1 := b [i][j];
      end;
      // Visible from right to left
      if (b [i][n + 1 - j] > max2) then begin
        mark [i][n + 1 - j] := True;
        max2 := b [i][n + 1 - j];
      end;
    end;
  end;


  // Find vertical visible books
  for j := 1 to n do begin
    max1 := 0;
    max2 := 0;
    for i := 1 to n do begin
      // Visible from top to bottom
      if (b [i][j] > max1) then begin
        mark [i][j] := True;
        max1 := b [i][j];
      end;
      // Visible from bottom to top
      if (b [n + 1 - i][j] > max2) then begin
        mark [n + 1 - i][j] := True;
        max2 := b [n + 1 - i][j];
      end;
    end;
  end;

  // Compute solution
  for i := 1 to n do
    for j := 1 to n do
      if (mark [i][j]) then
        sol := sol + 1;
  WriteLn (sol);
end.
```

Example solution in C#:

```
using System;
```

Day 1 – Task 2

```
class Books
{
	static void Main(string[] args)
	{
		string s;
		s = Console.ReadLine();
		int n = int.Parse(s);
		int [,] a = new int [n, n];
		for (int i = 0; i < n; i++)
		{
			s = Console.ReadLine();
			string [] numbers = s.Split(new char [] {' '});
			for (int j = 0; j < n; j++)
				a [i, j] = int.Parse(numbers [j]);
		}

		int sol = 0;
		for (int i = 0; i < n; i++)
			for (int j = 0; j < n; j++)
			{
				if (a [i, j] == 0)
					continue;
				bool visible = true;
				for (int k = 0; k < i; k++)
					if (a [k, j] >= a [i, j])
						visible = false;
				if (visible == true)
				{
					sol++;
					continue;
				}
				visible = true;
				for (int k = i + 1; k < n; k++)
					if (a [k, j] >= a [i, j])
						visible = false;
				if (visible == true)
				{
					sol++;
					continue;
				}
				visible = true;
				for (int k = 0; k < j; k++)
					if (a [i, k] >= a [i, j])
						visible = false;
				if (visible == true)
				{
					sol++;
					continue;
				}
				visible = true;
				for (int k = j + 1; k < n; k++)
					if (a [i, k] >= a [i, j])
						visible = false;
				if (visible == true)
				{
					sol++;
					continue;
				}
			}
		Console.WriteLine(sol);
	}
}
```