Clear all symbols from previous evaluations to avoid problems

```
Clear["Global`*"]
```

Percolation - square lattice

Hoshen-Kopelman algorithm for the square lattice, it returns a 2d array with all sites belonging to a cluster labelled by the same number

In[1]:=
```
HoshenKopelmanSquareLattice[lattice_] :=
  Module[{labelledLattice, n, m, lastLabel, leftLabel, labelAssignments, newLabels},
   {n, m} = Dimensions[lattice];
   labelledLattice = ConstantArray[0, {n, m}];
   labelAssignments = Array[# &, Ceiling[n * m / 2]];
   (* maximum number of clusters *)
   lastLabel = 0;
   Do[ (* loop over the lattice row by row *)
    If[lattice[[i, j]] == 1, (* if site is occupied then ... (otherwise move on) *)
     If[lastLabel == 0,
      (* if this is the first cluster, assign 1 *)
      labelledLattice[[i, j]] = 1;
      lastLabel = 1,
      (* otherwise *)
      If[i == 1,
       (* if we are in the first row,
       just copy the left neighbour (if unoccupied, it will be later corrected) *)
       labelledLattice[[i, j]] = labelledLattice[[i, j - 1]],
       (* otherwise copy first the label of the
        neighbour above (if unoccupied, it will be later corrected) *)
       If[labelledLattice[[i - 1, j]] > 0, labelledLattice[[i, j]] =
         labelAssignments[[labelledLattice[[i - 1, j]]]];
       (* and then for all columns but the first *)
       If[j > 1,
        (* check the left neighbour and possibly resolve conflict *)
        If[labelledLattice[[i, j - 1]] > 0,
         (* if the left site is occupied ... *)
         leftLabel = labelAssignments[[labelledLattice[[i, j - 1]]]];
         (* just to simplify things *)
         If[
          leftLabel < labelledLattice[[i, j]],
          (* if the cluster to the left has a smaller label then the one above,
             resolve conflict and replace all necessary labels *)
          labelAssignments =
           ReplaceAll[labelAssignments, labelledLattice[[i, j]] → leftLabel];
          labelledLattice[[i, j]] = leftLabel,
          (* else *)
          If[
           leftLabel > labelledLattice[[i, j]] && labelledLattice[[i, j]] > 0,
           (* if the cluster to the left has a larger label then the one above,
              resolve conflict the other way and replace all necessary labels *)
           labelAssignments =
            ReplaceAll[labelAssignments, leftLabel -> labelledLattice[[i, j]]],
           (* otherwise just copy the left label *)
           labelledLattice[[i, j]] = leftLabel
```

```
        ]
       ]
      ] (* if the left site is occupied *)
     ](* if we are not in the first column *)
    ]; (* if we are in the first row *)
    If[
     labelledLattice[[i, j]] == 0,
     (* if the label was not
      determined from neighbours then assign a new label *)
     lastLabel++;
     labelledLattice[[i, j]] = lastLabel
    ]
   ](* if it is the first cluster *)
  ], (* if site is occupied *)
  {i, 1, n}, {j, 1, m}
 ];
 (* relabelling *)
 newLabels = Array[# &, Ceiling[n * m / 2]]; (* maximum number of clusters *)
 lastLabel = 1;
 Do[
  If[
   labelledLattice[[i, j]] > 0,
   labelledLattice[[i, j]] = labelAssignments[[labelledLattice[[i, j]]]];
   If[newLabels[[labelledLattice[[i, j]]]] > lastLabel,
    lastLabel++;
    newLabels[[labelledLattice[[i, j]]]] = lastLabel
   ];
   labelledLattice[[i, j]] = newLabels[[labelledLattice[[i, j]]]]
  ],
  {i, 1, n}, {j, 1, m}
 ];
 Return[labelledLattice];
];
```

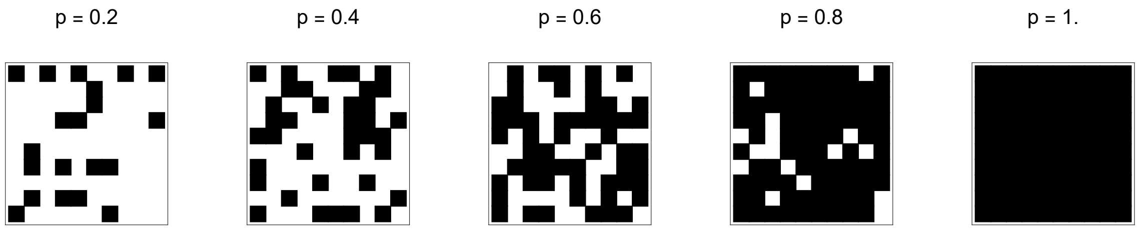
Function taking the result of the Hoshen-Kopelman algorithm for the square lattice and returning an array with the sizes of clusters

In[2]:=
```
SizesOfClustersSquareLattice[labelledLattice_] :=
 Module[{n, m, lastLabel, nclusters, clusterSizes},
  {n, m} = Dimensions[labelledLattice];
  clusterSizes = ConstantArray[0, Max[labelledLattice]];
  Do[ (* loop over the lattice row by row *)
   If[
    labelledLattice[[i, j]] > 0,
    clusterSizes[[labelledLattice[[i, j]]]]++;
   ],
   {i, 1, n}, {j, 1, m}
  ];
  Return[clusterSizes];
 ];
```

Example of square lattices 10 x 10 for various probabilities p

In[144]:=
```
{n, m} = {10, 10}; (* lattice size *)
np = 5;
lattice = ConstantArray[0, np];
clusters = ConstantArray[0, np];
probabilities = 0.0 + 0.2 Range[np];
Do[
  lattice⟦ip⟧ = Array[If[RandomReal[] <= probabilities⟦ip⟧, 1, 0] &, {n, m}];
  clusters⟦ip⟧ = HoshenKopelmanSquareLattice[lattice⟦ip⟧],
  {ip, 1, np}
 ];
GraphicsGrid[{Table[ArrayPlot[lattice⟦ip⟧, ColorRules -> {1 → Black, -1 → White},
    PlotLabel → Row[{"p = ", probabilities⟦ip⟧}]], {ip, 1, np}]}, ImageSize → Full]
Print["Hoshen-Kopelman labbeling:"];
Table[MatrixForm[clusters⟦ip⟧], {ip, 1, np}]
Print["Sizes of clusters:"];
Table[SizesOfClustersSquareLattice[clusters⟦ip⟧], {ip, 1, np}]
```

Out[150]=



Hoshen-Kopelman labbeling:

Out[152]=

```
{
 1  0  2  0  3  0  0  4  0  5      1  0  2  0  0  3  3  0  4  0
 0  0  0  0  0  6  0  0  0  0      0  0  2  2  0  0  0  4  4  0
 0  0  0  0  0  6  0  0  0  0      0  5  0  0  6  0  4  4  0  0
 0  0  0  7  7  0  0  0  0  8      0  5  5  0  0  0  4  4  0  7
 0  0  0  0  0  0  0  0  0  0  ,   5  5  0  0  0  0  4  4  4  0  ,
 0  9  0  0  0  0  0  0  0  0      0  0  0  8  0  0  4  0  4  0
 0  9  0 10  0 11 11  0  0  0      9  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0      9  0  0  0 10  0  0 11  0  0
 0 12  0 13 13  0  0  0  0  0      0  0 12  0  0  0  0  0  0 13
14  0  0  0  0  0 15  0  0  0     14  0  0  0 15 15 15  0 16  0

 0 1 0 2 2 0 3 0 4 0     1 1 1 1 1 1 1 1 0 1     1 1 1 1 1 1 1 1 1 1
 0 1 0 0 2 0 3 0 0 0     1 0 1 1 1 1 1 1 1 1     1 1 1 1 1 1 1 1 1 1
 1 1 0 0 0 0 3 3 0 3     1 1 1 1 1 1 1 1 1 1     1 1 1 1 1 1 1 1 1 1
 1 1 1 0 3 3 3 3 3 3     1 1 0 1 1 1 1 1 1 1     1 1 1 1 1 1 1 1 1 1
 1 0 1 0 3 0 0 3 0 0     0 1 0 1 1 1 1 0 1 1     1 1 1 1 1 1 1 1 1 1
 0 0 1 1 0 0 5 0 6 6  ,  2 0 0 1 1 1 0 1 0 1  ,  1 1 1 1 1 1 1 1 1 1  }
 0 1 1 1 1 1 0 0 6 6     0 1 1 0 1 1 1 1 1 1     1 1 1 1 1 1 1 1 1 1
 7 0 1 1 0 1 0 6 6 6     1 1 1 1 0 1 1 1 1 1     1 1 1 1 1 1 1 1 1 1
 7 0 0 0 0 1 0 6 0 6     1 1 0 1 1 1 1 1 1 0     1 1 1 1 1 1 1 1 1 1
 7 0 8 8 0 0 6 6 6 0     1 1 1 1 1 1 1 1 1 0     1 1 1 1 1 1 1 1 1 1
```

Sizes of clusters:

Out[154]=
```
{{1, 1, 1, 1, 1, 2, 2, 1, 2, 1, 2, 1, 2, 1, 1},
 {1, 3, 2, 12, 5, 1, 1, 1, 2, 1, 1, 1, 1, 1, 3, 1}, {20, 3, 13, 1, 1, 12, 3, 2}, {83, 1}, {100}}
```