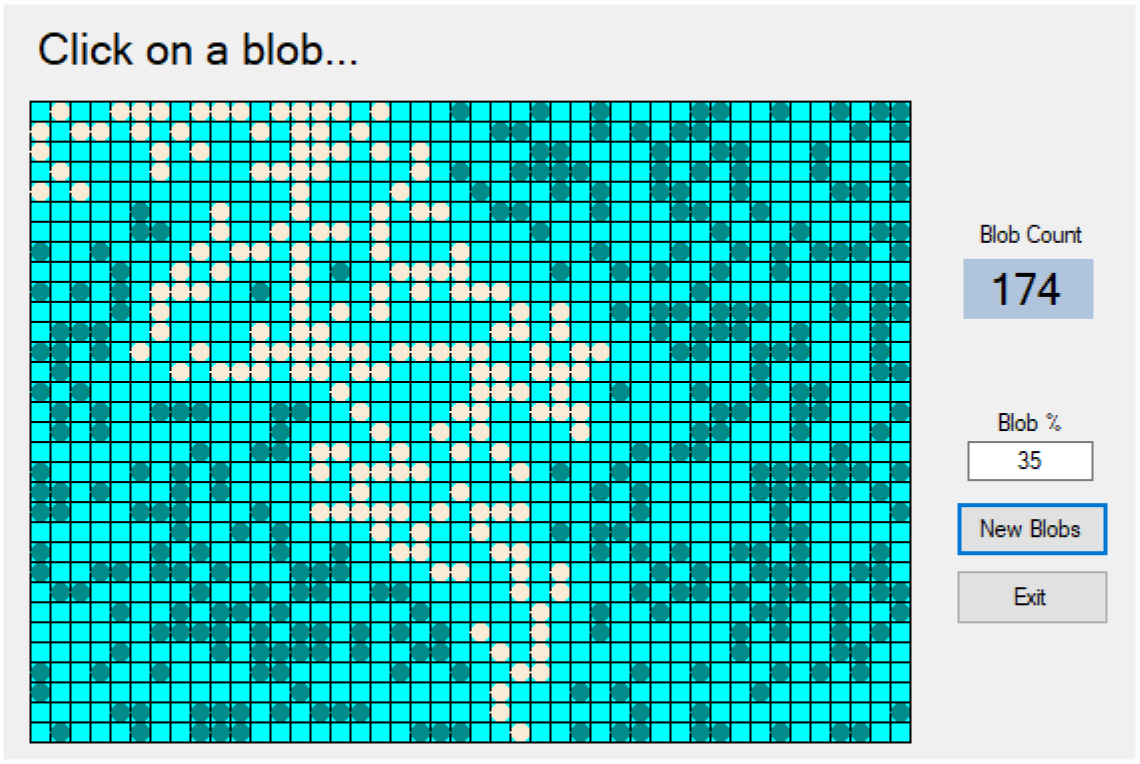
Blobs

Sometimes it is necessary to estimate the size of an irregular object. One way to accomplish this estimation is to overlay the object with a grid and count the number of cells that contain part of the object.

Consider a blob simulation program. Each cell of the grid is either empty or contains a piece of a blob. This determination is random, based on a percentage chance that a cell is occupied by a blob. The size of the blob is then determined by the number of adjacent cells, in every direction, that contain part of a blob. The finished product might look like this:



Assume that we are using a class named `PetriDish` that contains a 2-dimensional array of `Blob` objects. Each `Blob` object has a boolean variable that indicates if that piece of the blob has already been counted. Class `Blob` has the following methods:

`void setCounted(boolean TorF)`	-sets counted to true or false.
`boolean isCounted()`	- returns true if this blob piece has already been counted.

Class `PetriDish` has the following methods:

`int getNumRows()`	- returns number of rows in the grid.
`int getNumCols()`	- returns number of columns in the grid.

`Blob getBlobPiece(int row, int col)` - returns `Blob` object in the grid at `row, col`. Returns `null` if there is no `Blob` object at `row, col`.

Your task is to write the `PetriDish` method that returns the size of the blob at a specified location. Assume that the method will be called like this:

```
    . . . myPetriDish.countBlob(row, col)
```

Use this method declaration for your recursive solution.

```
public int countBlob(int row, int col)
{
```