2D arrays Lab

1) Declare and create a 2-dimensional array of floats named `numbers` and fill it with randomly generated values.
2) Modify your answer to 1) so that `numbers` is created as a 4-dimensional array of floats and fill it with randomly generated values.
3) Modify your answer to 2) so that the array `numbers` is created as a ragged 4-dimensional array instead. Only the last dimension needs to be ragged. Use random integers for the lengths of the ragged rows.
4) Modify your answer to 3) so that the array `numbers` is created as a ragged 4-dimensional array, and all dimensions are ragged. Use random integers for the lengths of all rows.

5) Consider the following method. Describe the value returned by a call to this method.

```
int mystery(int[][] numbers, int val){
  int idx = -1;
  for (int i = 0; i < numbers.length; i++){
    for (int j = 0; j < numbers[i].length; j++){
      if (numbers[i][j] > val){
        idx = j;
      }
    }
  }
  return idx;
}
```

6) Write a function `int maxSum(int[][] matrix)` which determines which row or column in the 2D array `matrix` has the maximum sum and returns it (the sum).
7) Write a function `int[][] transpose(int[][] matrix)` which returns the transpose of the input 2D array `matrix`. Recall that the transpose T of a matrix M is defined such that T[i][j] = M[j][i], for all i and j.
8) Write a function `PImage select(int x, int y, int s)` which takes an `x` and a `y` screen coordinate and returns an image that is `s` by `s` in size and contains the pixels that make up the `s` by `s` neighborhood around `(x, y)`. For example, `select(mouseX, mouseY, 10)` will return a `10` by `10` pixel region that surrounds the current mouse location. (In the case where s is even, there should be more pixels to the left and above the mouse position.)