Computer Science AP

Gridder – 2d Array Algorithm Practice

Open the project called 2DArrayProject.
You will be using the frame called Gridder for this practice sheet.

Preamble

This frame has code to draw the frame, squares, and control an automated running timer. You don't have to worry about that code. Just focus on the 2d array and using it to complete the tasks at hand!

The 2d array is called *grid* and is set to be 100X100. Find this near the top of the program.
The draw code currently assumes that the grid is set up using [column][row].
The grid is drawn with the following rules: black is 0 and white is 1. The entire grid is set to 0 when the program starts.
If you change the values in the grid, just call *draw()* to have the grid re-drawn. That's it.
You can select to draw in black or white and click to draw in the grid.

Example: Change the upper left corner square to white
```
grid[0][0] = 1;
draw();
```

Common Algorithms to Try

Code the following algorithms inside of jButtons 1-12. Test out!

1. When the button is pressed, select a random location within the grid and turn that square to write. Repeat this 100 times (use a loop, don't copy and paste 100 times)

2. When the button is pressed, select a random column. Set all the squares in this column to white. Then draw().

3. When the button is pressed, color the entire top half of the grid white and the entire bottom half black. You can use *clearGrid()* which is already coded in the project to help you in your answer.

4. When the button is pressed, count the number of squares in the grid that are white. Print out the total in the textbox *textInfo* with a line like:
   textInfo.setText(Integer.toString(count));

5. When the button is pressed, clear the grid and then create a white X (make a white line from top left corner to bottom right corner and bottom left corner to top right corner).

6. When the button is pressed, inverse the grid. All black squares will become white and all white squares will become black.

NOTE: A second 2d array is need to solve these next few problems. Here is an example problem. Please give some thought as to why a second 2d array is needed in order for this to work!

Example: The user will draw some white on the grid near the center of the grid. When the button is clicked, any square having a white square above, below, to the left, or to the right of it will turn white.

Solution:

```
//create temp
int[][] temp = new int[100][100];

//copy contents of grid into temp
for (int row=0; row<100; row++)
  for (int col=0; col<100; col++)
    temp[col][row] = grid[col][row];

//scan grid and MAKE CHANGES TO TEMP, not grid!
//stop and think about WHY we don't want to change the original grid.
//note: I will only check rows/columns 1-98 so my code won't break when I hit
//end edge!

for (int row=1; row<99; row++)
  for (int col=1; col<99; col++)  {
     if ( grid[col][row-1]==1 )            //check above
          temp[col][row]==1;
     if ( [col][row+1]==1)             //check below
          temp[col][row]==1;
     if ( grid[col-1][row]==1 )        //check to the left
          temp[col][row]==1;
     if ( grid[col+1][row]==1)        //check to the right

  }  //col
}  //row

//and finally, make grid equal temp and draw

grid=temp;

draw();

//smile
```

Not so bad. You can try running this code using *grid* in replacement of *temp* and see what happens.

Now try a few of your own. Remember not to be careful near the edges of the grid so you don't create out of bounds errors.

1. When the button is pressed, any black square touching TWO white squares will turn white. Consider all eight directions!

2. When the button is pressed, all squares will move one position to the left. The leftmost column will become the rightmost column. If you press the button fast enough, this should create a 'scrolling' effect with your drawing.

3. This last one is very few lines of code, but, seems to give students a difficult time each year. It's all about testing and planning this one on paper to find a simple pattern that exists.

   When the button is pressed, rotate the grid 90 degrees to the right. Most of you have seen this option in photoshop or photo programs where you want the entire image to rotate by 90 degrees. Well, get to it!

   Hint 1
   you need to use a second, temp grid
   Hints 2-5
   draw this on paper with a small grid (5X5) and do a rotation
   watch where a square starts and where it finishes
   watch all squares in a row and see how their column and row change
   watch all squares in a column and see how their column and row change
   LOOK FOR THE PATTERN – THERE IS A PATTERN.

   Once you have the simple mathematical pattern, start with a clear *temp* and then put the final positions of the white squares into temp.

   Don't give up on this one. It's very short code once you find the pattern.