3) (10 pts) DSN (Bitwise Operators)

A lights out puzzle is a classic puzzle played on a 2D grid that involves turning the cells of the grid off. The cells can be toggled by pressing a particular cell. Doing so will invert the cell pressed and the 4 adjacent cells that are the immediate neighbor (joined by a cell line). Your program stores a "lights out" puzzle represented using an array of integers, where each integer in the array represents a row of the puzzle. The bits when on represent that the cell is on, and when off represent that the cell is off. If in some row the first, sixth, and seventh cell is on, then the value at the spot in the array would be $97 = 2^0 + 2^5 + 2^6$. If we were to press the button in column 1 of this row, then it would turn spot 0 off, spot 1 on, and spot 2 on, changing the bit mask on that row to $2^1 + 2^2 + 2^5 + 2^6 = 102$. In addition, if the row above existed one bit would be changed there and if the row below existed, one bit would be changed there.

You will create a function that will emulate pressing a particular cell. You will be given the row and column (0-based index) of the cell toggled along with the grid and its dimensions. Modify the grid based on the described interaction. Be careful of indexing out of bounds. (Hint: There are 5 bits total that will potentially flip. One of those bits will always flip while the other four bits, the neighboring bits, will only flip if they are within the bounds of the grid.)

```
void cellPress(int row, int col, int height, int width, int * grid) {

   grid[row] ^= (1<<col);

   if (col)
      grid[row] ^= (1<<(col-1));

   if (row)
      grid[row-1] ^= (1<<(col));

   if (row < height - 1)
      grid[row+1] ^= (1<<(col));

   if (col < width - 1)
      grid[row] ^= (1<<(col+1));
}
```

Grading: 2 pts for each of the five sections of code. For the ones with the if statements, 1 point for the if and 1 pt for the toggle.

Note: Due to some confusion during the exam, give full credit for a valid 2D array solution where grid is two dimensional, the appropriate out of bounds checks are made and the corresponding XORs (all with 1 because presumably 0 or 1 would be stored in each cell of a 2D array) occur.