```c
#include <stdio.h>
#include <stdlib.h>

// Wrapped around neighbors
int modulo(int inp, int size){
    return inp >= 0 ? inp % size : (inp + size) % size;
}

int gridColour(int** grid, int pos, int size, int colours){
    if(pos == size * size){ // Base case
        for(int i = 0; i < size; i++)
            for(int j = 0; j < size; j++)
                printf("%d", grid[i][j]);
        printf("\n");
        return 1;
    }

    // Find the row and column of the next blank position
    int row = pos / size;
    int col = pos % size;
    int soln = 0; // How many solutions hereon
    int temp = 1;

    for(int i = 1; i <= colours; i++){
        int isLegal = 1; // Flag to check if color i can be placed here
        if(modulo(row - 1, size) == row)
            isLegal = 0; // Nope! I am my own neighbor
        if(modulo(col - 1, size) == col)
            isLegal = 0; // Nope! I am my own neighbor

        // Check constraints - can we place color i here?
        isLegal *= grid[modulo(row - 1, size)][col] != i;
        isLegal *= grid[modulo(row + 1, size)][col] != i;
        isLegal *= grid[row][modulo(col - 1, size)] != i;
        isLegal *= grid[row][modulo(col + 1, size)] != i;
        if(isLegal){
            grid[row][col] = i;
            soln += gridColour(grid, pos + 1, size, colours);
        }
    }
    grid[row][col] = 0; // Reset
    return soln;
}

int main(){
    int n;
    int c;
    scanf("%d %d", &n, &c);
    int **grid = (int**)calloc(n, sizeof(int*));
    for(int i = 0; i < n; i++)
        grid[i] = (int*)calloc(n, sizeof(int));

    int soln = gridColour(grid, 0, n, c);
    printf("%d", soln);
    return 0;
}
```